1 Introduction

Welcome to the Second Year Practical Skills course. This course is one component of an integrated practical skills programme extending through the four years of your degree. Generally speaking the *aim* of practical skills training is to equip you with those skills relating to laboratory physics which an employer would be likely to expect to find in a graduate in physics whether you are employed as a scientist or in a related field. To fulfil this aim the Department sets a series of *objectives* for the programme which are stated in Appendix A. The aim of the second year practical skills course is to fulfil some of these objectives as detailed below.

In fulfilling these objectives we intend that you should find the course interesting as well as being educational and informative. If you do not, we will consider this a failure on our part. In a later section you will learn how we ask for your help in giving us feedback on your experiences in the course. This is invaluable to the Department in making the course a more useful and relevant experience for you. We ask for your full co-operation in this process.

2 Safety in the Laboratory

- Safety is a paramount consideration in the design of the courses which run in Laboratory II. In general the experiments carry no greater level of hazard than you would encounter when using a domestic appliance except for a few cases where low level hazards particular to physics experimentation are involved (low powered lasers, weak radio-active sources etc). In these cases the scripts for the experiments will carry information and warnings as to the particular hazards and the safety regime to be followed to ensure safe practice at all times. You are expected to follow such safety instructions to the letter and both the supervisor and the lab technicians will check to ensure that you do so.
- All equipment used in the laboratory is regularly maintained and in particular mains powered electrical equipment is subject to testing to make sure that it conforms to required safety standards. Testing is carried out in the long vacation preceding the academic year. All electrical equipment used in the laboratory will carry a sticker indicating that this has been done and the date when. Visual checks are carried out during the term by the technical staff to spot equipment which has deteriorated in use. Nevertheless be vigilant in ensuring your own safety. If you notice a mains plug which is coming loose or mains lead which is cracking etc, please draw these to the attention of the lab technician who is instructed to deal promptly with such matters.
- In spite of the courses being designed with safety in mind, a laboratory can be an unsafe place if the occupants behave irresponsibly in it. Any such behaviour will be viewed seriously and can in extreme cases lead to disciplinary action.
- Note that maintaining a disciplined atmosphere and tidy environment in the laboratory is a prerequisite of safe practice. For this and other reasons:

  College Regulations prohibit smoking eating and drinking in the laboratory.

  Students are reminded that College Regulations also prohibits the use of mobile phones.

- Apart from hazards intrinsic to experimentation, the only general hazard which may affect you in the laboratory, as in other parts of the College, is that of fire. Should fire break out in the laboratory or the alarms sound to warn of fire elsewhere in the building then your ***only*** duty is cooperate in the speedy and orderly evacuation of the laboratory to the assembly point. The laboratory technician is the fire warden for the area and will, with help from the supervising staff, direct you through the fire exits at each end of the laboratory, down the stairs to the ground floor and out of the building to the assembly point in the Physics Yard to the South of the building. Cooperate at all times. Do not attempt to fight fires yourself and do not attempt to use a lift in leaving the building.
- Should an accident occur which requires medical attention, inform the supervisor or the laboratory technician who will take the initiative in summonsing a first aid trained person or other medical assistance.

3 Objectives

At the end of the course you should —

- have acquired increased skill and confidence in the acquisition of experimental data through the performance of experiments at the intermediate level;
- have extended the range of items of equipment used in a physics laboratory with which you are familiar;
- have improved your ability to record your work concisely and precisely as you perform it through repeated practice in recording experiments in your laboratory notebook, and with frequent feedback from teachers;
- have increased understanding and ability to apply the principles of data and uncertainty analysis introduced in first year to the more complex experiments performed in second year;
- have gained experience in the use of commercial software for the plotting and the statistical analysis of data;
- have increased ability to condense the information in your personal notebook record of an experiment into a concise but precise and complete formal report of the experiment in the appropriate style and format and suitably word processed;
- be familiar with, and able to use for the solution of scientific problems, a range of programming techniques in the QBasic language beyond the introductory concepts introduced in first year.

4 The Components of the Course

2B70 is a half unit laboratory-based practical course for first year students taking the part-time evening Physics B.Sc. degree.

The course is taken in **Terms 1 and 2** and has the following mandatory components:

- **Computing:** This follows on directly from the introduction to QBasic taken by students in first year. It consists of about six sessions in which each student works on one of the laboratory computers on a series of exercises (Units) guided by a teaching text. Each of the Units introduces a new element of QBasic programming technique. The course supervisor is on hand to help with individual difficulties. Each of the 6 Units has an assessed computer programming task. The detailed objectives for this part of the course are listed in Appendix B and the Units in Appendix C.

- **Physics Laboratory Experiments:** Experiments done in 14 three hour sessions cover the basic techniques of laboratory physics as well as illustrating some aspects of the physics topics taught in the lecture courses. The experiments fall into two groups; level 2 basic and level 2 extended (see section 5.4). The basic experiments are of the same standard as the extended experiments encountered in the first year laboratory and the list includes some which some in the class may already have done in that year. The level 2 extended experiments are of the same standard as experiments encountered in the third year laboratory course. All students are expected to attempt one of these and students who make sufficiently good progress may do more. A full programme on which 100% may be obtained for this section of the course consists of 10 experiments (extended experiments counting double) which must be recorded and reported as described below (Sections 7 & 8).
- **Formal Experiment Reports:** Full formal reports are to be produced for two of the Physics experiments that you do.

Marks are awarded by continuous assessment of each of the above components and count toward the final degree on the same footing as for any other half-unit course (see Section 10).

5 Organisation of Course

5.1 Timetable and Attendance

The course runs for 22 evenings during the first and second terms. Each student will attend for one three-hour session each week on either Tuesday or Thursday between the hours of 6 and 9pm working individually on experiments or computing exercises. You will be assigned to one of these evenings during the introductory lab session on the Tuesday of the first week of term. You may only attend on the assigned evening except under special circumstances and by prior arrangement.

- The course begins in Term 1 with six weeks of experimental work in the laboratory. This takes place in Laboratory I (south end of building, first floor). During this time you will work on experiments assigned from the basic list (see section 5.4).
- In the remaining six weeks of term, the computing course will run in Laboratories I and II making use of the laboratory computers on all of which QBasic is installed.
- In the second term, individual experiments are again done but based in Laboratory II. Students who have made sufficient progress will at this stage move onto level 2 extended experiments.

Note that the timing of the three parts of the course may be varied depending on class progress.

The Physics Laboratory will be closed during vacations and throughout the third term.

5.2 Computer Programming

During the last six weeks of the first term you will extend your working knowledge of the QBasic programming language by completing five tasks working from an instruction text (see Appendix C) and supervised by the course supervisor. Each one of the tasks introduces some new aspect of QBasic not met in the first year course. In the sixth and final period devoted to QBasic, you will attempt a task for the solution of which you will have to integrate the techniques learned in the five previous periods.

The five initial tasks are designed so that a student who successfully completed the QBasic course in first year should be able to complete each one in a single period. However for some students additional work outside the classes will be necessary. This can be done on the College computers in the cluster rooms (say, on the Tuesday or Thursday when you are not attending the computing or physics laboratory class), on a suitable computer at home, or on the computers in the laboratories. Normally you should not enter a laboratory to use the computers when a class is in progress. However it is quite all right to attend Laboratory II on the Tuesday or Thursday evening when you are not doing your physics laboratory work to work with the computers since there is a more than adequate number of machines. You can also get advice from the laboratory supervisor whenever he is not occupied with the students of the laboratory class. The final task is intended to require an additional five hours of work for its completion. If your schedule of completion of the set experiments is sufficiently advanced you may have time for this is in the regular laboratory periods. If not you will be expected to complete it using the facilities indicated above.

As you work through the Units of the QBasic course you will be required to save your work in a file on a floppy disk with which you will be supplied. It is these files which will be assessed at the deadline set for each Unit. You are not allowed to remove these classwork disks from the laboratory. In order that you may continue unfinished work in your own time, you will be supplied with a homework disk onto which you must copy, at the end of a the laboratory period, any task on which you wish to continue working. At the start of the next session you must copy onto the classwork disk any work you have updated. Only work on the classwork disks will be assessed. If you update a Unit on your classwork disk after the deadline for its completion, you must first inform the course supervisor. It will be subject to the penalty for lateness detailed in Section 5.6.

5.3 Physics Laboratory Staff

In the laboratory your work will be overseen by the laboratory class supervisor. Also present or on call will be the laboratory technician. The supervisor is there to provide advice and assistance to the students as well as allocating experiments and marking students' notebooks. At the start of each class the supervisor will go round to every student to ensure that they understand the experiment they are doing. Thereafter he will keep an eye on each student's progress, occasionally checking the notebook to see that the measurements are being made correctly and recorded and analysed properly. If you are uncertain about anything at any time you, should immediately let the supervisor know so that he can come to your assistance at the earliest opportunity.

It is perfectly all right to contact the technician directly about any matter relating to a particular piece of apparatus or the general use of the computers, but you should not expect technical staff to be able to assist with the experiment as such or to give detailed advice on using particular computer programs.

5.4 The Physics Laboratory Experiments

Each of the experiments has a title and an identification code. At the start of each laboratory session be sure to write *both* on a fresh page in your notebook before doing anything else making sure that you leave some space for any work required to complete the analysis of a previous experiment.

Here is the list of the experiments currently available.

LEVEL 2 BASIC

A10 X-Ray Diffraction
CP4 The Velocity of Light
CP7 Charge to Mass Ratio (e/m) of the Electron
CP8 Photo-electric Effect and Planck's Constant
CP9 Wavelength of the H_α line in the Spectrum of Atomic Hydrogen
CP2 Amperes Law
E12 Wheatstone Bridge
E13 Frequency Dependence of Inductance
E58 The Current Balance
H3 Ratio of the Principal Heat Capacities of Air
H8 Latent heat of Nitrogen
P3 Uses of an Oscilloscope
S5 Measurement of the Speed of Sound

LEVEL 2 EXTENDED

E7 Electrical Resonance
O5 Reflection of Polarised Light
CP10 Determination of the Quantum of Charge
M3 Mechanical Resonance

5.5 Laboratory Notebooks

During a Physics Laboratory class everything that you write down must be written directly in your laboratory notebook — nothing whatever should be written in any other notebook or on any separate sheet of paper unless specifically required for a particular experiment. This notebook must be of the correct type having alternate pages of lined paper and of millimetre or two millimetre grid graph paper. Suitable notebooks in either soft-backed or hard-backed versions may be purchased cheaply from the technician in Lab I. Section 7 of these notes deals with the manner in which these laboratory notebooks are to be used.

5.6 Deadlines and Penalties

The final deadline for all coursework is the Thursday of the first week of the third term and no work will be accepted after that day. Note that all work must be on hand by this date for second marking.

The **deadline** for handing in each Unit of computer programming course work is approximately five weeks after the material is covered in the course and is stated in each Unit handout. If you miss the deadlines you will lose credit on those assignments at the rate of 10% per working day. Each Unit will only be accepted for marking when all previous Units have been handed in.

The **deadlines** for the first and second formal reports are the Thursdays of the first and last weeks of the second term respectively. However you should note the advice in Section 8 counselling you to aim for earlier handing in. If you miss the deadlines you will lose credit on the assignments at the rate of 5% per working day.

Note that it is always worthwhile handing in work provided that this is before the final deadline. *Any student whose mark in the course falls below the pass mark solely because of late penalties will be credited with a pass in the course at 35%.*

Students who experience difficulty in meeting the deadline due to illness or the like should inform the Course Supervisor or the Course Tutor at the earliest opportunity.

There are no specific deadlines for the completion of lab notebook reports on individual experiments. Your normal progress through the course should see you completing experiments at regular intervals throughout the course. Failure to proceed in this way will only mean that work piles up unacceptably at the end of the course and may compromise your ability to meet the final deadline for handing in of all work.

Work should only be handed in to the Course Supervisor or the lab I technician who will date stamp it and record that it has been received.

6 Laboratory Procedures

6.1 The Student's Responsibilities

A Laboratory is potentially a dangerous place:

- Do make sure that you follow carefully any specific safety instructions issued with any experiment.
- Report anything which you consider might be dangerous to the lab technician.
- Familiarise yourself with the notes on laboratory safety (Section 2).

College regulations prohibit smoking, eating and drinking in the laboratory or the use of mobile phones.

Part of the first session will be taken up by an introduction to working in Laboratory II and what is expected of you. Thereafter experiments will be assigned at the rate of one per week by the laboratory supervisor. You will be given the script for the next week's experiment at the end of the current period. It is essential that you read through this before the next laboratory session to ensure a prompt start. You should also make every endeavour to turn up promptly on time at the start of each session in order to give yourself the maximum time in the laboratory. Experience has shown that this is the key to the successful completion of all aspects of the course.

When you first enter the laboratory collect your notebook and make your way to the bench where your experiment is set out. Read through your work of the previous week to see what comments have been made by your supervisor. Try to progressively improve the standard of your work based on this feed-back.

- Do not move apparatus from one bench to another without permission.

Read through the script once more and resolve any difficulties of understanding by discussion with the supervisor. Then begin the experiment as instructed.

Never turn on an electrical power supply or water supply or any light source until the supervisor has spoken to you.

In approximately the first six weeks of the course, when you are performing Level 2 basic experiments, you will not normally be allowed to take your lab notebooks from the laboratory. These will be retained, examined and commented on by the course supervisor for your guidance before the start of the next session

6.2 Performing the Experiments

The data taking and most analysis on experiments are designed to be completed to an *adequate* standard by an average student in one three-hour session (two sessions for extended experiments). This involves:

- recording in very brief note form the procedure and apparatus used;
- obtaining and correctly recording sufficient data of a reasonable quality;
- performing an adequate analyses on the data, including the drawing of graphs where appropriate and always including a quantitative estimation of uncertainties;
- writing a "Conclusion".

(See Section 7 for more details on keeping your notebook.)

Keep a check on the time and seek assistance if you experience any difficulties or feel you might run out of time. The supervisor is there to *help* you. You do not need to disassemble the apparatus when you have finished. At the end of the session your notebook will be collected by the supervisor. Your work will be marked during the week and the notebook returned to you at the next session.

In the first part of the course you will perform a new experiment every week, even if you do not manage to quite complete the one of the previous week. This is because you need to gain experience from a wide range of experiments in order to quickly build up your skills and knowledge of the techniques. In the later part of the course there will be time for you to return to the analysis of experiments that were not quite completed provided that you are regularly attending the class. However it is important that you always maintain the aim of acquiring an adequate quantity of data and carrying out a significant amount of analysis within one session, otherwise your work cannot be assessed and you will not learn. To simply make the measurements and then leave the laboratory in the hope of performing the analysis at some later date is a sure-fire recipe for failure. If your laboratory supervisor finds that you are accumulating a significant number of experiments lacking analysis, he will not assign you an experiment for one lab period. You will still be expected to attend on that occasion for the purpose of catching up on incomplete analysis work.

To progress satisfactorily, it is imperative that you attend regularly and work for the full three hours of each laboratory session.

Performing the more challenging experiments of Lab II is part of the long process, begun in first year, of becoming a competent experimental scientist. Be aware that you will encounter something new with each experiment so that you can steadily acquire and practice experimental and analysis skills. Do not worry therefore if your notebook is returned to you covered in red corrections — only try to understand where you have made mistakes and avoid repeating the same mistakes next time. This is the way to improve and gain a good grade in the course.

6.3 Instruction Sheets (Scripts)

Instruction sheets (the "script") are provided for all experiments. They are intended to define the purpose of the experiment and to give guidance on the experimental procedure and the methods of analysis to be employed. They do not *necessarily* contain all the information required for successful completion of the exercise; they must not be treated as a recipe to be followed exactly and without thought. Study all of the script carefully and thoughtfully *before* starting an experiment and seek advice from your supervisor if in doubt about anything. In this way the time spent working with the apparatus will be used most efficiently. Scripts also indicate which sections of the Physics course textbook you might refer to for additional theoretical background. The script should be kept by you either attached to the pages of the notebook or in a separate ring file.

When you are recording your experiment in your laboratory notebook do not copy out sections of the script, there is nothing to be gained by this. Do not regard the script as an example of a laboratory notebook or formal report to be imitated. The script fulfils quite a different purpose to either of these. Rather follow the directions given in the Sections 7 and 8 which explain in detail what is required.

6.4 Apparatus

You will encounter apparatus of a wide range of sophistication during the course. Many items have identification marks and you should note these down where relevant so that if you want to repeat some measurement at a later date you can do so under the same conditions; this is more important for the item being measured than for the measuring instruments themselves.

Many experiments use apparatus that must be connected to the mains electrical supply. Sometimes this involves plugging in to a nearby wall socket, but often the outlets mounted along the side of the bench must be used. These are connected by long cables to wall sockets, so ensure that the appropriate wall socket is switched on. As a safety precaution switch off all apparatus and the wall socket as soon as the experiment is completed. Do not switch on the power to an electrical circuit until it has been checked by the supervisor.

Light sources can be hazardous and some have special switch-on procedures, so do not switch them on until you have spoken to the supervisor. Caution is also necessary where liquids are concerned, so again do not proceed without having first consulted the supervisor.

Breakages and defects should be reported to the technician or the supervisor. If in any doubt on the use of any item of apparatus consult your supervisor. If apparatus appears to be faulty or if items are missing, inform the technician.

7 The Laboratory Notebook

7.1 The Aim

You should regard your laboratory notebook as a sort of diary in which is recorded your progress sequentially through an experiment. A laboratory record of any experiment must contain sufficient information to enable the experimenter to retrieve the work done on the experiment at a later date, either in order to write it up (e.g. as a formal report) without having to appeal to memory, or to allow someone else to repeat the same experiment to extend or confirm the observations made previously. A professional report should therefore contain a brief description of what has been done and the techniques and apparatus used, and should identify any difficulties encountered together with the ways in which these difficulties have been overcome etc. However, because your time in the laboratory is very limited, you are not required to duplicate material that is already contained in the experiment script. Do however make sure that it is clear from your record which part of the experiment you are carrying out at each stage, and note down anything that deviates from or is additional to what the script describes.

All numerical readings obtained must be recorded, even if subsequently discovered to be erroneous, and should never be erased. The deductions made from these observations, together with details of the way in which these deductions were made, should also be recorded. You should strive to acquire the attitude of a professional scientist who, at some later stage, will have to use this laboratory record to write up the results of the experiment, including possible re-analysis of the data, perhaps for publication.

The record must be entered *directly* into your notebook as the experiment proceeds. Do not keep notes and experimental data on loose pieces of paper or rely on your memory for subsequent writing up. Write down instrument readings *exactly* as they are taken from the instrument without any arithmetical manipulation. Clearly the notebook might not be as neat and tidy as you would wish. It may contain discarded data and erroneous calculations, but this will not be held against you provided such items are clearly labelled as being incorrect and the reason for identifying them as such is stated. Acquiring the ability to keep a clear and logical record of what is being done, whilst it is being done, is an essential part of a training in experimental physics. Your notebook will be used to form a judgement of your ability in experimental physics, and is therefore a very important component of the course assessment.

Check points:

- Write **only** in the notebook.
- Use only black or blue ink pens, not red or green, and **never pencil**.
- Never erase anything you write down: if something proves to be incorrect draw a single line through it and make a note of where in your notebook the corrected version is to be found.
- Number the pages of the notebook.
- Start each new experiment on a fresh lined page and leave space enough for your supervisor to add comments.
- Normally the graph paper pages should not be used for text.
- Do, of course, always aim for 100% legibility.

7.2 The Contents

A laboratory record should contain at least the following:

- **Identification:** Experiment code (e.g. P53), title of experiment (e.g. "Simple Pendulum"), the date.
- **Objectives**: A very brief statement of the objective of each stage of the experiment and the technique you will use. In most cases the experiment will be testing a hypothesis, in which case the relevant formulae should be quoted. Theoretical derivations of formulae given in the script need not be copied out (although a reference to where the derivation may be found would be useful). Nor do you need to give a lengthy description of the experimental procedure — just give the minimum necessary to enable another physicist, unfamiliar with the particular apparatus used but with the script available, to repeat what you have done.
- **Diagrams**: Sketches or diagrams of the apparatus and/or circuit diagrams are necessary for most experiments. The drawings in the script are designed to explain the theoretical background and give an idea of the purpose of the experiment. Your diagrams should express what you have done and how. Elaborate draughtsmanship is not necessary.
- **Measurements**: The observations made and readings taken should be recorded in a table with appropriate headings directly in the notebook. The formatting of the recorded values (number of significant figures) should always correctly reflect the precision of the experimental measurements. If there are calculations to be made upon each observation (and there frequently are) then design your table carefully to allow recording these computed values alongside the raw measurements. Avoid ever having to copy data from one table into another: this is a frequent source of error. The heading of each column of the table should consist of the name of the quantity, its mathematical symbol, and the units used. If a sequence of repeated measurements is made (e.g. micrometer readings) they must all be tabulated — not just their mean, for example. The credibility of your uncertainty analysis can be assessed, or the data re-analysed, only if the total raw data are presented.
- **Calculation of results**: You should always carry out at least a specimen set of intermediate calculations *whilst* the readings are being taken to ensure that the data is acceptable. Units must always be stated for the raw data and for any calculated quantity: do not think to yourself "it's obvious". Whenever you calculate a mean, always find the standard deviation and standard error at the same time, and write them down. Calculate the final result as soon as you have sufficient data to do so, so that any problems can be identified as early as possible. Underline and emphasise your key results: those that directly relate to the purpose of the experiment.
- **Uncertainties**: An assessment of the statistical uncertainties of the raw data must always be made at the time of making the measurement. The raw data uncertainties should then be appropriately combined and/or propagated through all intermediate calculations to arrive at an estimate of the accuracy of the final results. You must also try to identify likely sources of systematic uncertainty and, if possible, estimate their effects.
- **Graphs**: Graphs usually help to highlight and analyse results. Choose convenient scales that are easily interpolated and that allow sufficient accuracy. Where appropriate, plot the independent variable (the one directly under the control of the experimenter) as abscissa: e.g. fringe number, mass added to scale pan. Label axes with name, number and units of the quantity plotted.

  Mark data points clearly with for example a dot enclosed by a circle, a cross or other symbol. Indicate where appropriate the uncertainties of individual

measurement on the graph using error bars on each plotted point; seek advice about this if you are in doubt. In general, experimental points on a plot should not be directly joined up by line segments. A line appearing on a plot should represent some hypothesis that is being tested by the data, usually a single straight line that is the closest fit to the points. There are no hard and fast rules when presenting graphical data; the guiding principles should be clarity and ease of use. Graphs must be drawn on the proper pages at the relevant place in your notebook — loose pages are only acceptable when using special types of graph paper and should then be fixed into the notebook. *Always* plot graphs *as you proceed* — do not wait until you think that you have completed the data-taking to plot your data or results. In this way will you be able to judge as you proceed whether, for example, your intervals between data points are suitable or whether something has suddenly gone wrong.

- **Analysis**: Here the data is compared with some hypothesis. This can take various forms. If it is a simple matter of comparing a quantity calculated from a formula with a "standard" value then this can be done within the *Conclusions* section of your report. Often though you will have measured the same quantity more than once using different conditions or different methods, so you will need some calculations to illustrate how much variation there is and whether it is consistent with your estimated uncertainties. Another common situation involves analysing the relationship between two quantities as the conditions are varied progressively by the experimenter, and may require a straight line or other function to be fitted to a set of points already plotted on a graph. In the first year course you became familiar with the manual way of doing this involving drawing straight lines through the data by eye using a ruler. This remains a technique which may be used with Lab II experiments but you should also learn to use the "least squares" fitting and plotting programs maintained on the laboratory computers (Easyplot and Lsfit) to plot graphs and find gradients, intercepts and uncertainties. This is a statistically more meaningful way to estimate these parameters than the manual method. Instructions on the use of the programs is available in the laboratories and from the lab supervisor. You will be expected to present plots and fits produced in this way for your formal reports.

The following information is always placed in the ***Conclusions*** section at the end of your report. "Conclusions" is the accepted title for this section because it *concludes* the report, i.e. it comes last. Do not think of the term as meaning "deductions". The *Conclusions* section must be written in such a manner that it makes sense when read alone, without the script or the main body of your report. Hence all physical quantities must be referred to by name, not symbol ("the refractive index of a glass prism…" not "n = …").

- **Summary**: This is a quantitative statement of the main results, i.e. as numerical values with uncertainties and units. Regardless of where else they may appear in your experimental record, a clear summary of the final results and their uncertainties should be given at the beginning of this section. Some experiments will be measuring quantities whose values are listed in standard data tables (e.g. *Kaye and Laby*, available for consultation in the laboratory). When this is the case, the standard values should be given and compared to your results. This comparison is, of course, only meaningful when you are able to compare the discrepancy between your numerical result and an "accepted value" with your estimate of the uncertainty in your experiment.

- **Commentary**: This is a critical assessment of the experiment. For most experiments the principal element of this is a statement of the extent to which the results of the experiment agree with accepted values or support a

hypothesis or theory — unless something radically unexpected occurred during the performance of the experiment, and even then you need only comment on this very briefly. Remember any comparison with accepted values or theory can only be done by reference to the uncertainties on any measured quantities. Other things which may be worthy of comment include:

- Assumptions or approximations made.
- Internal consistency of the readings and/or the results.
- Comparison of the scatter of the results with the final random uncertainties assigned.
- Unexpected features in the data and their interpretation if possible.
- Possible sources of systematic error (this and the three above are interrelated).
- The apparent limitations of the apparatus used and/or the script and suggestions for improvement.
- Possibilities for improving the accuracy.

No laboratory record can be passed as satisfactory unless it has appropriate *Conclusions*.

7.3 Checking by the Supervisor

During the first part of the course when level 2 basic experiments are being done, you will not be allowed to remove your notebook from the laboratory. After each laboratory period it will be scrutinised and commented upon before the next week. The experiment which is assigned as the subject of the first formal report will be recorded in a separate laboratory notebook which will be provided. You will be allowed to take this out of the laboratory whilst you are preparing your formal report.

All laboratory notebooks MUST be handed in by the first Thursday of the third term for completion of any first marking and for second marking. Work that is not handed in promptly by this time may not be passed on to the Examiners.

7.4 Assessment of Laboratory Notebooks

Each experiment recorded in your laboratory notebook will be assessed according to these criteria:

- data of sufficient quantity and quality;
- data analysis correct and complete, producing the result required;
- uncertainties estimated for raw data and correctly calculated for results;
- quality of presentation and the writing of a properly headed *Conclusions* section.

The above criteria will be approximately equally weighted, though this will vary according to the experiment. Note however that should you omit any of the items in the above list from your record of an experiment, it may be deemed too fragmentary for any mark to be awarded. In particular no credit will be awarded for an experiment for which data has been taken and recorded in your notebook unless a minimum of analysis has been performed enabling a judgement of the quality of the data to be made. The analysis may, for instance, take the form of a graph.

8 Formal Reports

You will be required to prepare formal reports for two of the experiments which you have completed. The first will be set in the sixth week of the first term and must be handed in by the Thursday of the final week of term. The second report will be set in the sixth week of the second term to be handed in by the Thursday of the final week of that term.

The formal reports will be mostly prepared outside the lab using the notes already recorded in your notebook.

8.1 Purpose

The writing of a formal report can be considered as an exercise in the preparation of a scientific paper for publication. Whereas the essential feature of the laboratory notebook is the recording of all procedures and all results in such a way that the experimenter or an informed colleague could repeat the experiment or re-analyse its results at a later date, and which gives your teachers an insight into how you go about your work, the essence of the formal report is the communication of the experimental method and its key results to a wider audience. To emphasise the point, the lab notebook is a record of what you do, compiled as you make your way through an investigation and as such, particularly when you come to do longer experiments or projects in later years, may contain things you try which do not work out, thoughts which you jot down which in retrospect lead nowhere and the like. The formal report is a digest of this from which you have excluded irrelevant material in an effort to convey the essence of your investigation clearly, fully and without ambiguity. In similar vein a scientist preparing a report may have to add information not contained in his laboratory notebook in order to make the work more comprehensible or informative to a third party. Such items might be a potted version of the theory behind the technique used or references to allied work.

8.2 Content

The report should be complete and self-contained and written in good formal English with neatly drawn diagrams and graphs. In general the contents should include those items listed in Section 7.2 but with more detail. In addition, descriptions of the apparatus and procedures should be given in your own words. On no account repeat the script although you might make use of it: the script is not written as a report, being more akin to a series of instructions and is an unsuitable model. The report should be divided into suitable sections, each section having a clear heading. There are no strict rules about how this should be done, but a typical sequence of sections might be: *Introduction, Experimental Method, Experimental Data, Calculation of Results, Uncertainty Analysis, Conclusions*. If you start with such a structure in mind and vary it accordingly as you see that some element of your experiment does not fit into this pattern then you will soon become schooled into a habit of ordered, well structured presentation.

Two further elements need to be included to form a complete report. No report or scientific paper for publication is complete without an *Abstract* which is a short paragraph following the title and preceding the body of the report. The purpose of the abstract is to inform a browser or anyone conducting a search through the literature that the paper contains something which makes it worth reading. It is important that it arrest the attention of the reader. A good abstract is written in concise language and conveys three pieces of information: the purpose of the investigation, a statement of the techniques employed and a statement of the principal results. Finally a 'contents' page is most useful, especially in longer reports. It should be placed immediately after the abstract.

8.3 Style

The report should preferably be written in the past tense, passive form ("The heater voltage was then set to about 10V…"). Be sure to be consistent in your style of language and also in the style of page layout. Pay attention to how you lay out tables and equations, and note how roman characters used as symbols for physical quantities are conventionally printed in italic (e.g. current *I*) in order to distinguish them from ordinary language. In the laboratory there are examples available for consultation of well written reports and you are recommended to inspect these at some time during the course.

The text, tables and mathematics of both the formal reports for course 2B70 must be word-processed and the report presented as numbered sheets stapled or otherwise bound together. Graphs should be computer generated and any fitted straight lines labelled with slope and intercept. Note that if you use the lab package EasyPlot to do this then the graphs may be easily copied into documents prepared using the lab standard word-processing package, Word. It is an advantage if the diagrams are also computer generated but not a requirement. Hand drawn diagrams are acceptable but must be neat and inserted at suitable points in the text. All lab computers and the College computer system have the *Word* program which contains an *equation editor* for preparing mathematical formulae and an elementary drawing facility.

8.4 Assessment of Formal Reports

The formal report is regarded as primarily an exercise in scientific report-writing and members of staff try not to mark the notebook and formal report twice for the same thing, although some replication is probably unavoidable. You will however be penalised for errors which have been pointed out to you in your laboratory notebook relating to matters which can be corrected by the time you come to write your formal report (e.g. faulty propagation of uncertainties), but which you have nevertheless repeated in the formal report. The broad criteria for assessing a formal report are the following:

QUALITY OF THE ABSTRACT: all the elements of a good abstract should be present in concise form.

STRUCTURAL COHERENCE: the report should follow an ordered progression of ideas through the appropriate use of sections with informative headings. Pages, graphs, equations and tables of data should all be numbered and referred to by their numbers.

ENGLISH EXPRESSION: the report should be written clearly in concise grammatical English using correct syntax and punctuation.

PRESENTATION OF INFORMATION: the diagrams, graphs and tables should have titles and be neat and clear.

9 Computing

9.1 Misuse of Laboratory Computers

The laboratory computers are there primarily to support the practical work which takes place in the laboratory but also for students to use when preparing reports or solving problems associated with other parts of the degree. It would be an act of great discourtesy to fellow students, and very expensive of technical staff time to repair the situation, if you interfered with the programs or the system set-up in any way. The laboratory staff and supervisors maintain vigilance over the computers to try to ensure that this does not happen. Any interference which does take place will be viewed seriously and culprits identified will be subject to College disciplinary procedures.
On a lesser level, but still causing inconvenience, are sloppy or impatient habits in using the computers. You should not for instance create files on the hard disk. You will be issued with a floppy disk for the purpose of keeping your personal files. Similarly the response time of the computers, in particular in printout cycles, can be frustrating. Usually four computers share one printer. In these circumstances it is of no use becoming impatient if your job does not print out as quickly as you would like. In particular, sending repeated requests to print will merely cause the system to run even more slowly or even hang up completely.

9.2 Books

To help in the preparation of formal reports, you may find it helpful to have on hand the book recommended in first year, “WORD 6.0: A Progressive Course for New Users” by Coles and Rowley, published by DP, currently priced £6-95.

For the computer programming the HELP facility within QBasic is useful but can appear a bit cryptic and so a reference book is an advantage. A moderately priced book that is quite adequate is “MS-DOS QBasic" by K. Jamsa, published by Microsoft Press, currently priced at £6.95. If you have a computer at home, you may find that you already have this book, since some manufacturers supply it with the computer.

Unfortunately that book is now quite difficult to find in the shops and may be too concise for some people's taste. There are three other good books on QBasic that are currently available and provide a more “friendly” approach:

1. “Beginner's Guide to QBasic” by O. Melnikova, A. Bonushkina and V. Kryolov, published by Wrox, priced £19-95;
2. “QBasic by Example” by G Perry, published by Que, £27-49;
3. “QBasic Programming 101” by G Perry, published by SAMS, £26-95.

(1) and (3) include a computer disk containing example programs. (2) is a bigger book and more detailed than the other two. All three are written with the absolute beginner in mind. [See also the notes on computer programming.]

When buying any computing book check that it has an adequate reference section and a good index as well as any “tutorial” approach.

For the purposes of reference, you will find one or more copies in each laboratory of ;

1. “Table of Physical Constants” by Kaye and Laby

2. “Handbook of Chemistry and Physics”

3. “QBasic by Example” by G. Perry

9.3 The Computers

In the Physics Department laboratories on the first and second floors are various computers which are available for your use at any time during College teaching hours, provided that there is not a practical class running at the time. The computers in Lab I are all Pentium II PCs running the WINDOWS 98 operating system. Those in Lab II are Pentium III machines also operating under WINDOWS 98. All machines have QBasic and the lab data analysis programs, Easyplot and Lsfit, installed. These may be accessed by double clicking on the appropriate icon. These are the same computers that you will eventually be using to analyse the data that you acquire in your physics practical class (for more information see Appendix D).

In room D105(Physics cluster) are about 30 computers (IBM 486 running WINDOWS 3.1) which are available for use during the day and on Monday through Thursday evenings between 6 and 9pm. These computers are connected through the College network, just like those in other cluster rooms which are also available for your use. Currently there is little pressure on the cluster in D105 in the evenings and there are almost always machines available. However be aware that there is a booking system for all cluster machines and it is advisable to book a computer in advance, since a user who has booked has priority over a casual user.

9.4 Using the College PCs

To book a networked computer in advance so as to reserve it for your use go to the special booking system computer in any cluster room. Log in (as described below) and then follow the instructions.

To use the computers in D105 or any other cluster room you must "log in" at the beginning of each session. If the previous user of the computer has left it in the correct state you should find it with the *Logged Out prompt* displayed:
LOGGED OUT C:\⟩

To log in, type the word **login** (as a single word) followed by a space and your Username and then press the Enter key. You will then be asked by the computer to type in your password. Do this and then press the Enter key. After a short delay you will see the *Logged In prompt* displayed:
zcap*xxx* R:\⟩
where "zcap*xxx*" will actually be your own unique Username.

To do word processing with the *Word* program you first have to start up the system called *Windows*. Type **win** and then press the Enter key. After another short delay the graphical *Windows* screen will appear, with one of the icons (small pictures) named *Word6C*. Double click with the left mouse button on this icon to start the *Word* program (or alternatively single click on the icon and then press the Enter key).

To do computer programming with QBasic, at the *Logged In prompt* simply type **QBasic** and press the Enter key.

If you are using a floppy disk insert it after you have logged in. Remember to take it out again when you leave.

When you have finished using the computer, exit from *Word* and *Windows* or QBasic so that you have the *Logged In prompt* again and type **logout**. When you see the *Logged Out prompt* you can leave. Do not switch the computer off.

Note. To switch on a computer you generally have to switch on two things: the monitor or display unit and the system unit (box with slots for disks) that the monitor stands on.

Using the Printer in Cluster Rooms

To print your program you must first Exit from QBasic. Then at the DOS prompt type

print *progname***.bas**

where *progname* will in fact be the name of the QBasic program you wish to print. Note that you **do** need to include the ".bas" characters as part of the program's file name.

9.5 Using the Department PCs

Here you do not log in. These computers are normally to be found with the *Windows* system already running. To do word processing, click on the *Word* icon, as described for the cluster room computers. To access QBasic or the lab analysis programmes, Lsfit and Easyplot, double click on the appropriate icon to open the application. Alternatively you may access QBasic by exiting from *Windows* using the **File** menu, then typing **QBasic**, as described for the cluster room computers.

Note. You cannot access the network from the laboratory computers.

9.6 Using your own computer

If you have access to a suitable computer outside the College that has the appropriate software installed on it then of course you may use that machine to do your course work.

Provided the computer has the operating system DOS 5 or later it will almost certainly have QBasic installed. If you cannot find it on the PC's hard disk look on the original system disks or CD-ROM provided with the computer, or if the PC is at work ask the system manager to install the program for you.

Most IBM-compatible computers these days have the *Windows* operating system installed, but not necessarily the *Word* word-processor.

You must not copy any of these programs from another computer since they are commercial software and are therefore covered by copyright law.

9.7 Assessment of Computer Programming

The computer programming tasks will be assessed according to these criteria:

- appropriate explanatory comments through the use of REM statements;
- appropriate instructions for the user through the use of PRINT statements;
- a correctly functioning program;
- logical and simple program structure;
- appropriate choice of variable and disk file names;
- well-formatted output of results to screen and/or printer and correct use of disk input/output;
- Good use of graphics where appropriate to display program results;
- thorough testing of the program with sufficient samples of output.

10 Course Completion and Assessment

IMPORTANT:
All returned coursework should be kept safely by you and be made available for re-submission for second marking in the third term immediately following the Easter vacation. **Work that is not re-submitted at this time cannot be counted**.

Course assessment will be based on marks from all components listed in Section 4 of these notes.
For the computing component the marks for the first five units will count equally and contribute 65% of the credit for this part of the course. The final unit will count for 35%. The whole computing component will count equal to six experiments in the assessment of the course.

For the Physics Experiments component a mark will be formed from the best 10 of the experiments which the student has performed (each extended experiment counting as two). It is therefore advantageous for students to complete more than this number of experiments. The two formal reports will count equal to two experiments.

If no assessable work is received for any significant component, irrespective of the marks gained for the other components, the course will be deemed to be *incomplete*. Until work for the missing component has been handed in no course mark can be awarded.

Illness and other reasonable causes for non-achievement of the required work may be taken into account in the final assessment, but students who believe they have grounds for special treatment must inform the relevant Tutor and the Course Supervisor as soon as possible.

It is very important that you observe the deadlines which will be strictly adhered to. Work which is handed in late may well be refused.

Any student who feels that they are falling behind in their lab work should discuss the problem with the supervisor. Do not be tempted in such circumstances to copy results or analyses from other students. Such an offence is easily spotted and students are warned that they risk losing all credit for the experiment(s) in question; this applies both to the student who copies and to the one who has provided the material for copying. Serious cases will be referred to the College Tutor to Students for action under the College disciplinary procedures.

11 Student Evaluation of the Course

As in other courses, towards the end of the course you will be asked to fill in a "student assessment" form by which you may give your opinion of the course by assessing key indicators. The forms will be handed out in a lab period to be completed in the lab and passed on to a student volunteer who will collate the replies.

You will also find in the laboratory on the supervisor's table a supply of experiment assessment forms. These may be filled in on a voluntary basis if you wish to express any opinions about a particular experiment. The completed form should be posted into the box provided which is located on the supervisor's table. Both kinds of feedback are invaluable to the Department in trying to make sure that its laboratory courses run well and meet your needs. We ask for your co-operation in both exercises.

APPENDIX A

Aims and Objectives of the Practical Skills Programme

Aims

To equip the student with those practical skills and the flexibility of approach to the solution of practical problems which employers expect to find in Physics graduates employed as scientists in research and development or in other contexts.

Objectives

At the end of the full practical skills programme the student should:

- Be accustomed to working in an environment where best practice in matters of safety is required.
- Have developed observational, investigative and data analysis skills by the performance of a wide variety of experimental investigations.
- Have developed confidence and facility in the use of a wide range of technical equipment. In an era of rapid technological change this does not imply detailed training in the use of any particular equipment or technique. The student should rather have developed a receptiveness to the use of whatever equipment and techniques are appropriate for the solution of particular problems.
- Have increased conceptual understanding of some topics in the theoretical part of the degree through the performance of related experiments.
- Have developed enterprise, initiative and originality of approach in problem solving by taking part in project work.
- have acquired competence in oral and written scientific communication skills.
- Have developed skills in the use of computers within a variety of applications from standard packages used in various contexts and specialised programs used for the solution of particular classes of technical problem to self-motivated computer programming.

Note: The full extent of these objectives is met in a set of courses extending through all four years of the BSc (part-time regulations) Physics degree.

APPENDIX B

Objectives of the QBasic Component

At the end of the Level 2 QBasic course the student should :

- be more practised in using the techniques introduced in the Level 1 part of the QBasic course,
- be able to use OPEN statements to open variable length sequential disk files for output and input,
- be able to use PRINT and INPUT statements to write/read fixed length data records to/from disk files,
- be able to use the CLS statement to close a disk file previously opened for input or output
- to understand and be able to use appropriately dimensioned variables,
- understand the circumstances in which instruction loops are useful and be able to use the “FOR... NEXT loop,
- be familiar with “DO... UNTIL” and “DO... LOOP ...WHILE” looping techniques,
- be able to test for the end-of-file on a disk file
- understand the usefulness of structuring a program as a series of tasks each of which may be written as a subroutine,
- be able to write a subroutine procedure,
- understand the appropriate use of, and be able to use, the SHARED statement,
- be able with a SCREEN statement to set the computer screen in graphics mode and with a WINDOW statement delineate an area of it to be used for graphics,
- use a WINDOW statement to scale calculated co-ordinates onto the windowed area,
- draw points, circles and lines on the screen in a variety of colours using the PSET, CIRCLE and LINE commands,
- be able to integrate all the QBasic techniques learned in the solution of a scientific programming task.

APPENDIX C

QBasic LEVEL 2 COURSE

Introductory Remarks

The course is intended to continue where the Level 1 course left off. You should therefore revise carefully, as a preliminary to the Level 2 course, all the units you completed in Level 1. The **General Information** handed out in the Level 1 course remains pertinent to Level 2 and you should have it on hand for reference. The course will take place in Lab 2 and consist of you working through a series of units following an instruction text with an instructor (Dr Bartley) on hand to help you.

The Course Plan

The course is divided into several units, each with its own assessment task (except for the first, Unit 0), which cover separate topic areas. These are;

Unit 0: Preliminary unit
Unit 1: Recap of the topics in Level 1 QBasic
Unit 2: Reading/writing files from/to a disk. Dimensioned arrays
Unit 3: Looping and control
Unit 4: Subprograms and subfunctions
Unit 5: Graphics

Tasks and Assessment

Each of the units will be worked on during one lab period. If not completed in the period, you will have to complete the task outside of class. **The deadline for completion of each unit's task is five weeks from the class period in which it is set**. In Level 1 most of the set tasks resulted in printout to be submitted for assessment. In Level 2 you will submit completed tasks entirely on computer disks. You will have two, one to work on during the class period and which must not leave the lab, and one for you to take away to continue the work if necessary. At the end of each period you must copy the task in hand onto your homework disk if you wish to continue the work. At the start of the next period you must copy the continued task back onto the classwork disk if you have made any changes. When a task is complete, put a comment at the end of the program, "finished, task complete" adding also the date.
You will be given two disks, HOMEWORK and CLASSWORK. Each one will have two directories \DATA\ and \PROGRAMS\ for data files and program files respectively.

Copying files from HOMEWORK to CLASSWORK disks and vice versa:
The easiest way to do this is within QBasic. At the end of a class period when you have SAVEd the task onto your classwork disk in directory PROGRAMS, eject the disk and insert the HOMEWORK disk and SAVE it onto this disk by the same operation. At the start of the next class period, if you have worked on the task, insert first your homework disk and OPEN the file you have worked on from this. Eject the disk, insert the CLASSWORK disk and save the updated file onto this. You will get a message warning you that you are overwriting an existing file but this is what you want to do.

Work Identification

The first line of any programme you write must be a comment line giving your name, the unit concerned and the date on which you start the work.

What you are expected to know from Level 1

With Level 1 you covered some of the simpler aspects of programming in Qbasic. The following list is intended to jog your memory as to the most important concepts and QBasic statements you were introduced to. If they have faded from your mind you should go back and look through the assignments you did last year.

1. Numerical and string constants
   variable types (string, integer, floating point)

2. BASIC statements: keywords

3. REM
   CLS
   INPUT
   LINE INPUT
   PRINT
   LPRINT
   TAB
   LOCATE
   LET and =
   GOTO; labels:
   IF THEN
   IF THEN ELSE
   END

4. Functions ABS, ATN, SIN, COS, TAN, EXP, LOG, SQR

5. User defined functions: DEF FN

6. Subroutines: GOSUB, RETURN

QBASIC COMPUTING
UNIT 0

The following is a step-by-step guide through **unit 0's** tasks. In the unit you will learn, or be reminded of, the QBasic facilities; **SAVE AS**, **OPTIONS, SEARCH AND REPLACE, CUT AND PASTE, HELP (Index and Contents).**

Either from Windows or at the **DOS** prompt go into **QBasic.**

A window will come up which asks you if you want to see the "Survival Guide". To skip this or get out of the guide if you go into it, just **press <esc>**.

Go into the File menu and choose "**OPEN**". Insert your classwork disk into the drive and choose drive **A**. Go into the subdirectory "**DATA**" and choose the file "**WEEK1.BAS**".

This file is in the wrong directory. **Using "SAVE AS" from the File menu place it in the "PROGRAMS" directory.**

OK, you are now ready to start.

You will be interested only in the area of the program between the *starred* lines. Note the lack of numbered lines! You can get around the file by using the mouse or the arrowed keys.

First, go into "**OPTIONS**" and set the "**Syntax Checking**" option to be on.

Replace what is between the quotation marks for NAME$ with your name.

Use the "**CUT**" and "**PASTE**" utilities to move the indicated line down the program to where it is indicated you should place it. Return to your original position and continue.

From the "**SEARCH**" menu use the Find option to look through the program for the number "122" and change it to "120". Follow the instructions given.

Using the **HELP Index**, find and put a statement where indicated which will pause the program for two seconds (HINT: look under "S").

Use the immediate window to calculate the given equation (note that one number should have been corrected already!). Put your value in the next line for "y".

Using the **HELP** Contents, find the **ASCII** code for a **"BEL"** and then write a line setting "x" to this value, i.e., "X= "

Finally, write two lines of code to determine the area of a rectangular plate and the uncertainty in the area. If you need to know an intrinsic function, go to **HELP**.

RUN THE PROGRAM!

If you are successful *strange* things should happen. In the File menu **SAVE** your program. Eject your classwork disk, insert your homework disk and save the program onto this also. If this is the end of the session, **EXIT** from **QBasic** and hand in your classwork disk. If not continue to unit 1. The task will not be assessed but you should nevertheless attempt to finish it if it is incomplete at the end of the period since you will find familiarity with the items covered an advantage in the remainder of the course.

QBASIC COMPUTING
UNIT 1

Revision of Level 1 QBasic

In experiment H3 two lengths, h_1 and h_2, are measured with uncertainties δh_1 and δh_2. The ratio of the specific heats of air (at constant pressure, C_p ,and constant volume, , is calculated from these as; $\gamma = \frac{C_p}{C_v} = \frac{h_1}{h_1 - h_2}$. The calculation of the uncertainty on the value of γ is tricky (why?). A reasonable way to approach the problem is to first calculate the change in γ produced by a change of h_1 to $h_1 + \delta h_1$. That is, calculate

$$d\gamma_{h_1} = \left(\frac{h_1 + \delta h_1}{h_1 + \delta h_1 - h_2} \right) - \left(\frac{h_1}{h_1 - h_2} \right)$$

Similarly calculate the change in γ produced by a change of h_2 to $h_2 + \delta h_2$;

$$d\gamma_{h_2} = \left(\frac{h_1}{h_1 - h_2 - \delta h_2} \right) - \left(\frac{h_1}{h_1 - h_2} \right)$$

The uncertainty in γ, $d\gamma$, is obtained by adding the two part uncertainties in quadrature;

$$d\gamma = \left[d\gamma_{h_1}^2 + d\gamma_{h_2}^2 \right]^{/2}$$

The programming task is to write a program which will ask you to input the measurements and their uncertainties via the keyboard and print out the value of γ and its uncertainty on the screen. Write the program in the stages indicated below which will test your competence at using the elements of QBasic which you met in Level 1.

First enter the QBasic program screen by double clicking on the QBasic icon from WINDOWS

Stage 1.

Use the REM statement (or equivalent) to start with a remark stating the purpose of the program.
Use the CLS statement to clear the output screen.
Use the INPUT statement with suitable prompt strings to ask for the values of the measured quantities, h_1 and h_2, and their uncertainties to be entered and store them in variables called h1, dh1 and h2, dh2.
Insert CLS to clear the screen again.
Add PRINT statements to output, with suitable explanatory text, the values read in.
Use Locate to position the output at line 3.
Add an END statement to end the list of instructions (it is always good practice to end a program this way)
SAVE your program on your disk in the \PROGRAMS| directory (on the A: drive) AS UNCinG1.bas. RUN the program entering 31.0cm for h_1 and 5.4cm for h_2 with both having uncertainties of 0.07cm.

Stage 2

Comment out the PRINT statements of stage 1 (using ')
After the INPUT statements insert arithmetic statements to calculate γ, the partial errors $d\gamma_{h_1}$ and $d\gamma_{h_2}$ and the uncertainty on γ, $d\gamma$.

Use PRINT statements to output the calculated values of γ and $d\gamma$ and the input values of h_1 and h_2 to the screen. Make sure you include in the statements text to explain the output.
SAVE your program AS UncinG2.bas and RUN it inputting the values given for stage 1 as input. You should calculate the values $\gamma = 1.2109$ and $d\gamma = 0.0034$.

Stage 3

Use GOTO (with a labelled statement) to make the program run in an endless loop returning to the beginning after the calculation on a set of input data has been completed and printed out to ask for another set.
SAVE your program AS UNCinG3.bas and RUN it for the 3 sets of input data:
31.0, 5.4: 15.0,3.6: 10.0, 2.4cms all lengths having uncertainties of .07cm. You should obtain the three sets of values (for γ and $d\gamma$); 1.2109 ± 0.0034, 1.3158 ± 0.0084; and 1.316 ± 0.013 (Note; rounded to two significant figures in the uncertainty and same number of decimal places in the result).

Stage 4

The only way to break out of the program of stage 3 is to press Ctrl+c and EXIT QBasic. Re-enter from WINDOWS in the usual way. The program must be modified to provide a means of normal exit when all data has been inputted. There are a variety of ways in which this can be done. Try the following (or another if you can think of one and know how to code it). Before the GOTO statement add an INPUT statement asking " Any more data". Store the answer (y for yes, any other character to mean no) in a string variable. Use an IF .....THEN plus a logical test on the string variable to branch back to the labelled statement at the start of the program if the answer is y (for yes) or to the END statement if any other answer.
SAVE your program AS UNCinG4.bas and RUN it on the data of stage 3 exiting normally after all three data sets have been entered.

Stage 5

The code to calculate the uncertainty on γ is a self contained set of instructions to carry out a specific task. It could be separated off as a subroutine. Do this creating a subroutine called UNCER (and using GOSUB and RETURN). Note that in this case separating off the code into a subroutine is not very useful since the calculation is carried out at only one point in the program. Subroutines are most useful when the same calculations are carried out at several points from which the subroutine may be called avoiding repeating several times the same lines of code.
SAVE your program as UNCinG5.bas and RUN it on the same data as given in Stage 3.
When you think you have finished go back and make sure that your program has appropriate comments to explain what is being carried out at all points in the code.

End of Task: Copy the program files you have created to your homework disk and hand your classwork disk to the class supervisor before you go.

Deadline for task completion: Five weeks from the date of the lab session in which the unit is started.

QBASIC COMPUTING
UNIT 2

The topics to be covered in this unit include input/output from/to files on a disk and storage of data in dimensioned variables (arrays).

INPUT from Disk

Suppose we have a series of numbers stored on a disk which we want to read in. They will be stored in a string one after the other which is terminated by an end of file (**EOF**). The file will be in some directory and have a name which is just a character string say, in this case, **task2in.dat**. (There are some rules for naming files you should stick to. are given in the next section.). The computer is informed that this file is needed for input by the **OPEN** statement in the following form.

OPEN "a:\data\task2in.dat" FOR INPUT AS #1
The character string in "" indicates that the data is on the disk in the **A:** drive in directory **\data** and has the name **task2in.dat**. The #1 indicates that in the program to follow this data file will be referred to as file **1**.

To read in the data an **INPUT** statement is used in the following form;

INPUT #1, a,b,c,d,e
a,b,c,d,e are variables we wish to read the 5 first numbers into. Note if we only write **INPUT #1,a,b,c** only the first three numbers will be read off the file. But if we put two **INPUT** statements following each other like

INPUT #1, a,b,c,d,e
INPUT #1, f,g
then this will read the first five numbers and then the next two into f and g.
If you try to read more numbers than are in the file then you will read the end of file mark and an error will be signalled.

When you have finished reading all the data required from a file it should be closed with a **CLOSE** statement, in this case;
CLOSE(1), the number indicating that file **1** is to be closed .

File names

A file may be given a name up 8 characters long containing no spaces. Not all characters on the keyboard may be used but you will be safe if you stick to letters and numbers plus $ and _. It is best to think up a name which is related to the nature of the data or purpose of the program the file contains. In addition a file name may have an extension up to three characters long and separated from the name by a full stop (**.**). The extension is used to indicate what type of information the file contains. Thus files containing programs in **QBasic** always have the extension **.bas.** Files containing data in this course will always have the extension **.dat**. Thus the following are valid file names;

Week_1.bas contains a basic program. Note the way _ is used to separate a label (1) from the name. This is useful if you have a series of similar files, for instance;
Week_2.bas,
Week _3.bas,

Unit_1.dat in this course might be used to store data ,

Accounts.lis might contain a list of particular data from program **Accounts.bas.**

OUTPUT to Disk

To output to a file a similar procedure must be followed. The file must first be opened for output. Thus if we want to write the file the name **Task2out.dat** into the directory **\data**\on the **A:** drive use the statement;

OPEN "A:\data\task2out.dat" FOR OUTPUT AS #2
The file will be referred to in the program as file **2**. So if we want to output three quantities, a,b,c to the file a **PRINT** statement is in the following form;

PRINT #2 ,a,b,c
When you have finished writing to the file close the file using **CLOSE**, in this case **CLOSE(2).** This will write an end of file(**EOF**) after the data.

Arrays and Dimensioned Variables

In your programming until now, you have stored single items of data (numbers, character strings etc), each variable having a different name (A=3, ch$= "two" etc). Sometimes when dealing with several pieces of data (numbers, character strings etc) on which similar operations are to be carried out it is useful to store them not in variables with completely different names like a,b,c,d,e etc but as elements in a one dimensional array. Thus instead of a,b,c,d,e we could have variables A(1),A(2),A(3),A(4),A(5). The computer must be told that A is an array with 5 elements. This is done with a **DIM A(5)** statement best placed at the start of the program. In the program any element of the array might be referred to as A(i), where i=1,2,3,4 or 5.
Sometimes it is useful to store numbers (or any other variable type) as elements in a two dimensional array. Thus array B(3,3) can be thought of a s series of boxes arranged in 3 rows and 3 columns in which we might store the numbers 1 to 9 thus.

1 2 3

4 5 6

7 8 9

Generally the elements of B() are referred to as B(i,j) where i=1,2 or 3 and labels the row the element is found in, and j=1,2 or 3 and labels the column. In this example B(2,2) (row 2, column 2) contains the number 5 and B(1,3) (row 1, column 3) the number 3 If numbers are stored in a two dimensional array the computer must be told how many elements there are by using the **DIM** statement again best put at the beginning. In this case **DIM B(3,3)** declares that the array B has $3 \times 3 = 9$ elements.
You will realise that the arrays A(5) and B(3,3) are similar to what you would refer to in mathematics as matrices.

Task

The task is to calculate the point of intersection of two lines in two dimensions (solve two simultaneous equations) using line coefficient data on disk and outputting the result to disk. Follow the stages below remembering to comment your program as it develops to indicate the function of each part as well as the complete task.

Stage 1

On your disk in directory **\data** there is a file called **cline.dat** which contains the line coefficients. **OPEN** the file and **INPUT** the first three numbers from this file and store them in variables called, **c11, c12, c13**.
PRINT out the numbers to the screen.
INPUT the next three numbers, store them in the variables **c21, c22,c23** and similarly **PRINT** them to the screen. You have finished reading in the numbers so **CLOSE** the file.
SAVE the program **AS intrsec1.bas** in directory **\programs** on your disk and **RUN** it..

Stage 2

The six numbers read in are the coefficients of two simultaneous equations (straight lines in two dimensions);
c11x+c12y=c13
c21x+c22y=c23
These have the solution;

$$x = (c13 \times c22 - c12 \times c23) / (c11 \times c22 - c12 \times c21)$$

$$y = (c11 \times c23 - c13 \times c21) / (c11 \times c22 - c12 \times c21)$$

Use these equations to write arithmetic statements to solve for the co-ordinates x and y of the point of intersection.
PRINT the co-ordinates x and y to a file called **xya.dat** on your disk in the **\data** directory.
Don't forget to **OPEN** and **CLOSE** the file. **PRINT** the values of x and y to the screen with suitable text also.
SAVE the program as **intrsec1.bas** in directory **\programs** on your disk and **RUN**

Stage 3

The coefficients of the straight lines could just as well be stored as the elements of an array **Coeff(2,3)** (2 rows, 3 columns=6 elements).
Change your program throughout to use the elements of **COEFF** in the calculation rather than the variables **C11, C12** etc. (Hint: you will find the **SEARCH** and **REPLACE** facility very useful for doing this). Don't forget the **DIM** statement
Modify your program to store the results of the calculation (x and y of the intersection point) in the two elements of a one dimensional array **XY(2)** (don't forget to put in a **DIM** statement for this). Again the **SEARCH/REPLACE** facility will be useful for this.

SAVE the program as **intrsec2.bas** in directory **\programs** on your disk and **RUN**

Stage 4

Dimensioned variables need not contain only numbers. Any variable type can be stored but the array must have the correct form of name (just as a variable must have the correct type). Thus a name ending in $, like Name$(10) will be a one dimensional array with ten elements which can store 10 string variables. Modify your program to store the names of the input and output files in the two elements of array FILE$(2). (Do not

forget the **DIM** statement. Modify your program change the explicit names of the input and output files in the **OPEN** statements to elements of the array **FILE$(2).**
SAVE the program as **intrsec2.bas** in directory **programs**\ on your disk and **RUN**.

End of Task: Make sure your program is adequately commented. Copy the program files you have created to your homework disk and hand your classwork disk to the class supervisor before you go.

Deadline for task completion: Five weeks from the date of the lab session in which the unit is started.

.

Further note on arrays.

Arrays/Dimensioned variables are used a lot in programs for storing many pieces of data on which the same calculation has to be performed and for storing the results of these calculations. You will see how useful this is in the next task where structuring programs to perform repeated tasks (LOOPING) is addressed.

QBASIC COMPUTING
UNIT 3

In this unit you will be introduced to statements which enable parts of a program to be executed repeatedly in loops. You will also revise IF THEN statements. Together these enable you to control the way a program is executed.

LOOPING

It is often useful to be able to repeat a calculation or operation within a programme several times (in a loop) or until some condition is met when the program jumps out of the loop and continues. QBasic supplies several ways of doing this. We will look at just two by means of which you will be able to write code for most situations.

FOR .... NEXT Loops

This is most useful when you know how many times you wish to repeat a set of programme instructions. The simplest form of the statement is;

```
FOR   i=1 to 10
          .
          .
          .
  program statements
          .
    NEXT i
```

The program executes the statements between the **FOR** line and **NEXT** with first the counter i equal to 1. When NEXT is reached i is increased by 1 and the loop executes again with i=2. The looping continues with i increased by 1 each time until it reaches the value 10. Note that there is nothing magic about 10. You can set this number to anything and the loop will be performed until i reaches the value you have set. Nor does it have to be explicitly a number. You can write

FOR i=1 to num

where num is set or computed earlier in the program.

Example a: Suppose we want to compute, and print on the screen, the squares of the first 12 integers. The following will do this.

```
FOR j=1 to 12
  square=j^2
PRINT "The square of " ;j; "is";square
NEXT j
```

Example b: Suppose we want to read in 14 numbers from the disk file **a:\data\numbers.dat** and calculate and output their cubes to the screen.

```
OPEN "A:\data\numbers.dat" FOR INPUT AS #1
FOR i=1 to 14
INPUT #1, num
```

```
cube=num^3
PRINT " The cube of " ;num; "is";cube
NEXT i
```

Ending FOR NEXT loops early

You might, for instance, want to end computing the cubes in Example **b** if you encounter a number whose cube exceeds 144. The loop can be ended before I=14 using an **IF THEN** and an **EXIT FOR** statement thus;

```
OPEN "A:\data\numbers.dat"  FOR INPUT AS #1
FOR i=1 to 14
INPUT #1, num
cube=num^3
IF cube>144 THEN EXIT FOR
PRINT " The cube of " ;num; "is" ;cube
 NEXT i
```

TASK: Stage 1

In the **\data** directory on your disk there is a file of 26 numbers called unit3a.dat. Write a program to read in these numbers and calculate the sum of the square and cube of each one, outputting the number, and the sum to the screen. Terminate the calculation as soon as you reach a number whose cube is bigger than ten times its square.
Don't forget to comment your programme. **SAVE** it **AS** file **A:\PROGRAMS\taskau3.bas** and **RUN** it. Check manually that your program has terminated on a number which satisfies the condition.

DO UNTIL.... LOOP and DO..... LOOP UNTIL loops

In the loop above **EXIT FOR** and **IF THEN** combined give a means of controlling how often a loop is performed. QBasic has other neater means of doing this, the one we will illustrate being **DO UNTIL .... LOOP.** As the name implies the loop is executed until some condition is reached. It works as follows. Example **b** can be coded alternatively as;

Example c

```
OPEN "A:\data\numbers.dat"  FOR INPUT AS #1
DO UNTIL cube >144
INPUT #1, num
cube=num^3
PRINT "The cube of " ;num; "is";cube
LOOP
END
```

The program reads in the numbers in turn until it encounters one whose cube exceeds 144 when it terminates the loop.
Notice that the condition in the **UNTIL** is tested before any instructions in the loop are executed. There is a variant of this type of loop which tests the condition at the end of the loop so that the instructions in the loop are always executed once. This is the **Do....LOOP UNTIL** form.
Thus Example **c** could be coded as ;

Example d

```
OPEN "A:\data\numbers.dat"  FOR INPUT AS #1
DO
INPUT #1, num
cube=num^3
PRINT " The cube of " ;num; "is" ;cube
LOOP UNTIL cube >144
END
```

In this simple example it makes no difference when the test is performed but it can be important so you should know that the two forms of the loop exist.

END OF FILE (EOF) testing

The code in examples **b** and **c** could give trouble. What would happen if no number in the input file satisfied the condition? All numbers in the file would be read in one after the other and the input statement would try to read the next one which does not exist. It would in fact try to read the **EOF** mark and an error would be signalled. This can easily be avoided by using a **DO....LOOP UNTIL** loop (or the alternative form) and the **EOF** function**. EOF(n)** returns "true" (or "yes") when the record just read in a file **n** is the End of File mark and the value "false" (or "no") otherwise. Each time an input is read **EOF** can be checked to see if this is the last record on the file.
Thus we can write a loop reading in data from file 3

```
i=0
DO
i=i+1
INPUT #3 num(i)
 LOOP UNTIL  EOF(3)
```

This will read in each piece of data in turn into the next element of array num() until an **END OF FILE** is reached.

TASK: stage 2

Add to your code for stage **1** a test to check for **EOF** as well as the condition 'cube greater than 10 times the square'
SAVE the program **AS** file **A:\PROGRAMS\taskau3.bas** and **RUN** on first the data file **A:\data\unit3a.dat** and then on **A:\data\unit3b.dat**. In the latter there is no number which satisfies the condition. Print out a comment to the screen signifying this condition when it is found.

REMINDER

Don't forget when writing logical statements you can combine tests. Thus

```
IF a>b AND c=d THEN GOTO top
```

demands that a is greater then b and at the same time c equals d before the program will follow the GOTO direction to the labelled statement top:.

Ending DO..... LOOP UNTIL loops early

Suppose in Example **d** you want to end computing the cubes if a negative number is read in. This can be accomplished with the **EXIT DO** command as follows**;**

Example e

```
OPEN "A:\data\numbers.dat"  FOR INPUT AS #1
DO
INPUT #1, num
IF NUM<0 THEN EXIT DO
cube=num^3
PRINT "The cube of " ;num; "is"; cube
LOOP UNTIL cube >144
END
```

DO .....LOOP WHILE and DO WHILE .....LOOP loops

These are analogous to the **DO UNTIL....LOOP** and **DO.....LOOP UNTIL** forms where the loops are performed **UNTIL** some condition becomes satisfied. In the **WHILE** forms the looping continues as long as (**WHILE!**) some condition is satisfied. Use this form instead of the **UNTIL** form where it is appropriate.

Task : Stage 3

In file **A:\data\unit3c.dat** there is a list of numbers. Write a program to read them into an array numbers(100) (there are less than 100 numbers on the file). Sort through the numbers and find the largest and print it out (to the screen) with a suitable text string. Quit if a number which is negative is read in and in this case print out the largest number encountered up to this point together with a message indicating that a negative number has been encountered and its value. **SAVE** your program **AS A:\PROGRAMS\taskbu3.bas** and **RUN**

End of Task: Make sure your program is adequately commented. Copy the program files you have created to your homework disk and hand your classwork disk to the class supervisor before you go.

Deadline for task completion: Five weeks from the date of the lab session in which the unit is started.

QBASIC COMPUTING

UNIT 4

In this unit you will be introduced to a method for making long programs more understandable. The aim is to break a program up into modules which each perform a specific task. A program structured in this way is much easier to follow and if you ever have to change it, much easier to do so. You have already encountered one technique for doing this using the subroutine with the **GOSUB** and **RETURN** statements. The present unit introduces you to a more general way of doing it via **subroutine *procedures***.

Subroutine procedures

Go back to the program you wrote in **unit 1**. In this you coded a subroutine to calculate the error on γ. You simply copied some of the statements from the program you had written below the **END** statement of the program of **stage 4** and inserted the label **UNCER:** before them and added a **RETURN** statement at the end. This made a separate module of the error calculation. To perform the calculation you used **GOSUB UNCER** in the main module in place of the copied code. Note that all the variables inside the subroutine were the same as when the code was part of the main module. The subroutine is still really part of the main program you have written. If you had a more complicated program where this same calculation was performed more than once but on differently named variables it would be awkward to use the same subroutine code as above because this uses a particular set of variable names.
The **subroutine *procedure*** is one which gets around this limitation and enables the same calculation to be performed on different variables in different parts of the program.

The approach of breaking a program down into logical modules which can coded as **subroutine *procedures*** can be seen in the following example.

Example 1. Given four points in space (four sets of x,y,z co-ords) which define the vertices of an irregular tetrahedron we have to calculate which of the faces has the largest area and which the smallest.
Logically the program to perform the calculation can be broken down like this.

1. Read in the 4 sets of x,y,z

2. For each face calculate the area

3. Decide which face is the biggest and print out the result

4. Decide which face is the smallest and print out the result

The area of four sides must be calculated. To avoid repetition of calculations which are identical each time but performed on different data the calculation can be carried out in a **subroutine *procedure.***

This could be effected with the following program which uses the subroutine **AREA** (you need not try to understand the detail of the calculations in **AREA**).
Notice that each time the area has to calculated, the appropriate subroutine has to be **CALL**ed. The variables needed for the calculation (co-ords of vertices of the tetrahedron) have to be given to the subroutines in the form of a list in brackets following the subroutine name and the results of the calculation (the length of a side or area of a face) are given back to the main module via a variable in the same brackets.

Example 1 program

```
'This is the main module
'The program calculates the area of the largest and smallest faces of an irregular tetrahedron
DIM x(4),y(4),z(4), face(4)
'Input the co-ords into the elements of the arrays x(4),y(4),z(4)
FOR i=1 to 4
PRINT "Input co-ords of  point"; i
INPUT "x="; x(i)
INPUT "y="; y(i)
INPUT "z="; z(i)
NEXT  i

'Now calculate the area of each face and store in the elements of face(4)
CALL AREA(x(1),y(1),z(1), x(2),y(2),z(2), x(3),y(3),z(3),face(1))
CALL AREA(x(1),y(1),z(1), x(2),y(2),z(2), x(4),y(4),z(4),face(2))
CALL AREA(x(1),y(1),z(1), x(3),y(3),z(3), x(4),y(4),z(4),face(3))
CALL AREA(x(2),y(2),z(2), x(3),y(3),z(3), x(4),y(4),z(3),face(4))
'Decide which side is biggest and which smallest
biggest=face(1)
smallest=face(1)
FOR k=2 to 4
IF face(k)>biggest THEN biggest=face(k)
if face(k)<smallest Then smallest=face(k)
NEXT k
PRINT " The biggest side has area"; biggest
PRINT " The smallest side has area"; smallest
END

SUB AREA(a,b,c,d,e,f,g,h,i, size)
'Calculates the area of the triangle with vertices (a,b,c),(d,e,f) and (g,h,i)
L1=SQR((a-d)^2+(b-e)^2+(c-f)^2)
L2=SQR((a-g)^2+(b-h)^2+(c-i)^2)
costheta=(((a-d)(a-g)+(b-e)(b-h)+(c-f)(c-i))/(L1*L2)
sintheta=SQR(1-costheta^2)
size=(L1*L2*sintheta/2)
END SUB
```

********** If you want to run this program, it is on your classwork disk ********** as A:\PROGRAMS\UNIT4.BAS**

Notes:
a) The code for a subroutine always starts with **SUB** ***name of subroutine()***
and ends with an **END SUB**. After executing the last statement the program goes back to the statement in the main module after the one from which the **CALL** was made.

b) Subroutines are always placed after the **END** statement of the main program module. The **END** statement **MUST** be there. If not the main module will not stop there but continue into the statements of the subroutine!
c) The variable names used inside the subroutine are purely local and mean nothing in particular in the main program. The values needed for these variables are transferred to the subroutine by listing the main program variables whose values we wish to make them equal to in brackets after the subroutine name in the **CALL**. Sometimes it is useful and economical to use the same variable with the same meaning in both the main module and a subroutine. If you do this the subroutine has to be told (so it knows where to find the variable). This done using the **SHARED** statement. **SHARED** variables do not have to be put in the brackets after the subroutine name. For example, in the example above, the statements in the **FOR....NEXT i** loop reading in the data could easily be put in a subroutine **READIN.** In this case, instead of listing all these variables in brackets thus;
CALL READIN (x(1),y(1), z(1), x(2),y(2), z(2), x(3),y(3), z(3), x(4),y(4), z(4)),
the variables can be declared *in the subroutine* to be **SHARED** and the call would simply be
CALL READIN

The complete program would look like;

Example 2 program
'This is the main module

'and the area of the largest face of an irregular tetrahedron

```
DIM  x(4),y(4),z(4),face(4)
'Input the co-ords into the elements of the arrays x(4),y(4),z(4)
CALL READIN
.
.
'The succeeding statements are as in the original program
..
.
END

SUB READIN
SHARED  x(),y(),z()
FOR i=1 to 4
PRINT "Input co-ords of  point"; i
INPUT "x="; x(i)
INPUT "y="; y(i)
INPUT "z="; z(i)
NEXT i
END SUB

SUB  AREA(a,b,c,d,e,f,g,h,i,  size)
code as in example 1
END SUB
```

NOTE : the variables x(4),y(4),z(4) are properly dimensioned in the main module. The brackets in the subroutine indicate this. The number of elements in x etc need not be explicitly given in the subroutine.

Coding Programs with Subroutine Procedures

QBasic has special **EDIT** facilities to help you with this. Suppose in the above example you had just typed the first line of the subroutine **AREA** after the **END** statement, i.e. the line

SUB AREA(a,b,c,d,e,f,g,h,i, size)

As soon as you press enter the screen will clear and only the two following lines will appear;

SUB AREA(a,b,c,d,e,f,g,h,i, size)

END SUB

QBasic has entered the second line which must end every subroutine for you. You have then to enter the rest of the code in between. QBasic also adds a **DECLARE** statement to the main module for each subroutine you write. Don't be concerned by this. You need to understand **DECLARE** for more advanced programming but not at this level so just ignore its presence.

Editing subroutines and the main module

If, after trying to run the program, you want to correct mistakes you may have entered in the subroutine, or change some lines, then you can move up and down to the lines with the shift keys. You cannot get to lines in the main module if you have a subroutine on the screen or vice versa by the same procedure, using the shift keys only enables you to move within a module. If you need to move to another then select the **SUB**s option from the **V**iew menu. The subroutine names will be displayed. Use the shift keys to highlight the one you want and the press return (or use the mouse pointer to move the highlight and click on the one you want.) The screen will immediately display the item you want.

Task

You are given the position of 5 points in a plane in polar co-ordinates (r,θ) with in θ degrees. The task is to;
a) calculate the areas of the triangles formed by any pair of the vectors from the origin to each of the points and to print out to the screen the area of the triangles with smallest and biggest areas. The area of the triangle with the three vertices defined by the origin and the two points (r_1,θ_1) and (r_2,θ_2) is given by;

. $$Area = (r_1 r_2 Sin(\theta_2 - \theta_1))/2$$

The five points given are $(1,0^0),(1,45^0),(2,90^0),(1,135^0),(2,150^0)$.
Note that when you use the Cos function in QBasic the angles must be in radians.

Stage 1: Write a main module program to input the co-ordinates of the five points from the keyboard via a subroutine **INDATA** and store them in some suitably dimensioned arrays. **SAVE** your program **AS Maxmin1.bas** in the **\PROGRAMS** directory on your disk and **RUN** it. Do not forget to comment it to explain its function.

Stage 2: Add a subroutine called **TRIANGLE** to calculate the area of the triangle with the origin and any two of the points as vertices. **CALL** it repeatedly from the main program module to calculate the areas of all possible such triangles and store the areas in some suitably named array. **HINT**: work out how many triangles there are before you start to code the program so you know how many calls to **TRIANGLE** to make. Try to put the **CALL**s in **FOR.... NEXT** loops to shorten the code. Output each area to the screen. **SAVE** the program as **Maxmin2.bas** in **\PROGRAMS** and **RUN** it. Note the maximum and minimum values outputted.

Stage 3: Add a subroutine called **Maxmin** which searches through the values of the areas and picks out the minimum and maximum values. Output them with explanatory text to the screen. **SAVE** the program **AS Maxmin3.bas** and **RUN** it to confirm that it picks out the correct maximum and minimum values. Before you have finished the task make sure you have added informative but brief comments to explain the function of each part

End of Task: Copy the program files you have created to your homework disk and hand your classwork disk to the class supervisor before you go.

Deadline for task completion: Five weeks from the date of the lab session in which the unit is started.

QBASIC COMPUTING
UNIT 5

In this unit you will find out how to do simple graphics within your QBasic program. Any program action which results in output on the screen is a graphics action. Any such action requires the computer to be set in a particular **SCREEN MODE** of which there are twelve available within QBasic. In a particular **SCREEN MODE** the screen is divided up into a certain number of rows and columns of pixels each one of which can be set by a program instruction to be one of a list of colours. For instance when you write to the screen with a **PRINT** statement the computer is automatically set into **SCREEN 0** mode. This divides the screen up into 28 rows with 80 pixels per row, each one being the size of a character and each one of which can be set to one of 4 colours with a **COLOR** statement in the program. You have not made use of this but might care to try it by inserting the statement **COLOR 1** (or 2,3 or 4) in any of the programs you have so far written. To do much else requires the screen to be divided up into a greater number of pixels (more rows and columns) which can be set at a wider range of colours. The more pixels the screen is divided into, the more resolution to the pictures you draw. How many pixels the screen can be divided into depends on the capabilities of the computer and monitor you are using. The computers in the lab (and cluster room) are capable of supporting the screen divided into 480(rows) by 640 (columns) of pixels each one of which can be set to one of sixteen colours.

The computer is set to address the screen divided in this way by including the command **SCREEN 12** in your program. (The other **SCREEN** numbers support different numbers of pixels and ranges of colours and will not be discussed in this course.)

Drawing points: Setting pixels to particular colours

For numbering the pixels QBasic adopts a co-ordinate system with origin (0, 0) in the top left hand corner of the screen. Pixel (x,y) is in column x (across the screen) and row y (down the screen). The statement

PSET (x,y), n

included in a program (after **SCREEN 12**) will cause the pixel at position (x,y) to bright up with a particular colour. x must be no more than 640 and y 480. n which selects the colour is any integer from 0 to 15. The range of colours set by the 16 values of n is as follows.

n	0	1	2	3	4	5	6	7	8
Colour	Black	Blue	Green	Cyan	Red	Magenta	Brown	White	Grey

n	9	10	11	12	13	14	15		
Colour	Light Blue	Light Green	Light Cyan	Light Red	Light Magenta	Yellow	Bright White		

The pixel may be switched off again (cancelled) with the statement

PRESET (x,y)

or simply with

PSET (x,y), 0 since n=0 means no colour or that of the background screen (black).

Drawing lines from one pixel to another

The statement

LINE (x_1, y_1) **-** (x_2, y_2)**, n**

Causes a line to be drawn from pixel (x_1, y_1) to pixel (x_2, y_2) in colour n.

If you want to erase the line, use

LINE (x_1, y_1) **-** (x_2, y_2)**, 0**

since n means no colour.

Drawing Boxes

This uses a modified form of the **LINE** statement.

LINE (x_1, y_1) **-** (x_2, y_2)**, n, B**

Will draw a rectangular box with corners at (x_1, y_1) , (x_2, y_1), (x_2, y_2) and (x_1, y_2) in colour value n

Drawing circles of a given radius, r, centred on a particular pixel, (x,y)

The statement

CIRCLE (x,y), r, n

causes a circle of radius r (in pixel widths) to be drawn centred on the pixel in column x and row y in colour n. The statement

CIRCLE (x,y), r, 0

will erase the circle.

Example 1

To help you understand the above, load in the program **UNIT5Ex1.bas**. from the **\PROGRAMS** directory on your classwork disk. This is listed below. RUN it and then modify it as instructed in by the remarks in the program between the **********s **SAVE** and **RUN** the program at each stage to make sure you understand its function

```
' This is UNIT5Ex1.BAS
' Program to play with graphics
'Draw some lines and circles and then erase them
CLS
SCREEN 12
'Draw some points
PSET (100, 150), 2
SLEEP 1
'*******************************************************
'Add a line of code drawing a point at (200,250) in colour n=3
'*******************************************************
SLEEP 1

'Draw some circles
CIRCLE (150, 150), 10, 2
SLEEP 1
```

```
'*********************************************************************
'Add a line of code drawing a circle at (250,250) radius 20 in colour n=4
'*********************************************************************
SLEEP 1

'Draw some lines
LINE (100, 150)-(150, 100), 7
SLEEP 1
LINE (100, 150)-(150, 200), 10
SLEEP 1
'*********************************************************************
'Add two lines of code drawing lines from (200,250) to (250,200) in colour 8
'and from (200,250) to (250,300) in colour 11
' 1 sec delay between each line
'*********************************************************************
SLEEP 1

'Put a box around everything in white
LINE (10, 10)-(440, 440), 15, B
SLEEP 1
'*********************************************************************
'Add a second box 10 pixels bigger all round in yellow
'*********************************************************************
'Start erasing after 2 secs
SLEEP 2

'Erase points
PSET (100, 150), 0
SLEEP 1
'*********************************************************************
'Add a line of code erasing the point at (200,250)
'*********************************************************************
SLEEP 1
'Erase the circles
CIRCLE (150, 150), 10, 0
SLEEP 1
'*********************************************************************
'Add a line of code erasing the circle at (250,250)
'*********************************************************************
SLEEP 1
'Erase the lines
LINE (100, 150)-(150, 100), 0
SLEEP 1
LINE (100, 150)-(150, 200), 0
SLEEP 1
'*********************************************************************
'Add two lines of code erasing lines from (200,250) to (250,200)
'and from (200,250) to (250,300)
' 1 sec delay between each line erasure
'*********************************************************************
SLEEP 1

'Erase the box
LINE (10, 10)-(440, 440), 0, B
SLEEP 1
```

```
'*****************************************************************
'Add a line of code erasing the bigger box added (10 pixels bigger all round)
'*****************************************************************

END
```

Choosing an area of the screen to draw on and scaling to that area

Choosing a screen area: Graphics does not have to use the whole area of the screen. A portion of it can be reserved using the **VIEW** statement. Thus

VIEW (30,20)-(200,200)

reserves the rectangular area between the pixel co-ordinates (30,20), (30,200),(200,200) and (30,200)

The statement comes with two useful options in the form

VIEW (20,20)-(200,200),m,n

Selecting **n** to be one the integers 0 to 15 draws a border in the colour with that **n** value around the area and selecting **m** to be one the integers 0 to 15 shades the inside in the colour set by the value of **m**.

Scaling: The program may be calculating x and y co-ordinates over some range where x lies between x_1 and x_2 and y between y_1 and y_2. This range may be scaled to fit on the pixel area chosen with **VIEW** by using the **WINDOW** statement thus;

WINDOW (x_1, y_1) **-** (x_2, y_2)

Thus **WINDOW (-1,-2)-(1,2)**

means that the rectangular pixel area defined by the pixel co-ordinates (30,20), (200,20), (200,200) and (30,200) is now marked out in x and y co-ordinates with x running from -1 to +1 and y from -2 to +2. Now when in the program we calculate x and y points may be set and lines drawn in these co-ordinates directly by using **PSET(**x,y**)** and **LINE** (x_1, y_1) **-** (x_2, y_2)**, n** where the x and y are as calculated (i.e. not in pixels).

Adding Text

In **SCREEN 12** the statements **LOCATE** and **PRINT** can be used to write on the screen. The screen is divided into 80 column and 30 rows of character size pixels on any one of which a character can be written with **PRINT** (and **LOCATE** to position the text) It is tricky to decide where to position the text with respect to the "picture" graphics. Remember when you write **LOCATE**, each character in terms of **SCREEN 12** pixels is 8 pixels wide and 16 in height. You can choose a colour for the text (on black background) with the **COLOR n** statement before the **PRINT**.

Example 2

To help you to understand the above **OPEN** the file **UNIT5ex2.bas** in the directory **\PROGRAMS** on your classwork disk. **RUN** the program to see what it does. The program listing follows below. Understand what it does and then add statements as instructed between the **********s at the end of the program. **SAVE** and **RUN** the program at each stage to make sure you understand its function.

```
'This is UNIT5eX2.BAS
' Generates and displays on the screen three cycles of a Sine curve

CLS
'choose screen 12
SCREEN 12

'select part of the screen. Change the background colour
'and outline the selected area in bright white
VIEW (20, 20)-(300, 200), 2, 15
'change scale to x,y to match the size of the sine curve plotted
WINDOW (-1, -11)-(22, 11)
'Box the plot to clear the sine curve all round.
'Note the size of the box in y depends on the amplitude seebelow) of the curve
'and in x on the angular range to be covered (see below, 0 to 6pi)
LINE (0, 10.5)-(-10.5, 21), 15, B

' The program now plots the curve y=A Sin(x), A, the amplitude is
'10 in this case and x the angle. x will run fom 0 to 6pi radians (3 cycles)
'draw the x,y axes
'x axis
LINE (0, 0)-(22, 0), 15
'y axis
LINE (0, -10)-(0, 10), 15
'choose the spacing in x of the points to give 30 points on
'the curve per cycle
dx = 6.284 / 30
'calculate the y values for each x over 3 cycles (x from 0 t0 6pi radians)
FOR i = 1 TO 91
x = (i - 1) * dx
y = 10 * SIN(x)
' Plot each point and circle it
PSET (x, y), 5
CIRCLE (x, y), .2, 14
NEXT i

'Now add a heading to the curve
COLOR 2
LOCATE 3, 29
PRINT "Y=ASin(x)"
LOCATE 4, 30
PRINT "Sine"
LOCATE 5, 31
PRINT "curve"

'Add the new statements here as instructed below
'*********************************************************
'Using the programme above as a model
'Add a statement to select with VIEW the screen area from
'pixels(320,200) to (620,400).
'Outline it in bright white and change the background to
'some colour other than black.
'*********************************************************
'*********************************************************
'You will be drawing a cosine curve with amplitude 20 over three cycles
'on this area. Using a WINDOW statement as above scale the graph of the
'curve suitably to fit onto the screen area defined by VIEW
```

```
'************************************************************
'************************************************************
'Outline the area using LINE in some colour different from that used above
'************************************************************
'************************************************************
' Draw the x and y axes for the cosine curve using LINE
'************************************************************
'************************************************************
'Use a FOR NEXT loop as above to trace out the curve, plotting 20 points
'per cycle. Plot each point with PSET and CIRCLE each one as
' above but choosing different colours
'************************************************************
'************************************************************
'Add the heading..... Cosine curve..... to the graph
'************************************************************
END
```

Task

The equation of a spiral in polar co-ordinates (r,θ) is $r = a\theta$ where a is a constant. In this task it is set equal to 1. The task is to plot the spiral for values of θ from 0 to 6π radians.
Remember the relationship between cartesian and polar co-ordinates is;

$$x = rCos\theta$$
$$y = rSin\theta$$

Stage 1

Use **SCREEN** and **VIEW** commands to choose a rectangular area about half of the screen positioned in the middle of the screen to plot the curve on. Decide on the maximum and minimum values of x and y to be plotted and use the **WINDOW** command to scale the spiral dimensions onto the chosen area.
Picking suitable colours use the **LINE , , ,B** statement to draw a box around the plotting area and **LINE** to draw x and y axes of suitable length from the x=0,y=0 origin. **SAVE** the program in the **\PROGRAMS** directory on your classwork disk **AS Spiral1.bas** and **RUN** it.

Stage 2

Calculate x and y values for points on the spiral for values of θ from 0 to 6π radians. within a **FOR... NEXT** loop. Calculate 50 points per turn. Plot them using **PSET** with a different colour for points on successive turns of the spiral. **SAVE AS Spiral2.bas** in the **\PROGRAMS** directory and **RUN it.**

Stage 3

Use **COLOR**, **LOCATE** and **PRINT** to label the figure as “Three turn Spiral” at some convenient position on the figure. Label the axes as x and y and add scale markers at x=5,10,15and y=5,10,15. **SAVE AS Spiral3.bas** in the **\PROGRAMS** directory and **RUN it.**

End of Task: Make sure your program is adequately commented. Copy the program files you have created to your homework disk and hand your classwork disk to the class supervisor before you go.

Deadline for task completion: Five weeks from the date of the lab session in which the unit is started.

QBASIC COMPUTING
UNIT 6

So far in this course you have been introduced to a number of computing structures and practised these by writing short programs. The final task is to write a more extended program which will require you to use most of these structures to solve the complete problem.

The Task

You will now write a program to analyse data adapted from experiment H15, *The Thermocouple* which is not one you will encounter this year. In this experiment the Seebeck emf (E) generated in a Cu/Fe thermocouple is studied.

A chromel/alumel thermocouple is used as a thermometer for measuring the temperature (t) of the Cu/Fe thermocouple working junction during the experiment. The E-t relation for the chromel/alumel thermocouple is linear over the range 0 to 500 °C. Its temperature, and so the temperature of the Cu/Fe thermocouple, is given by:

$$t = \frac{E - E_0}{E_{100} - E_0} t_s \qquad [1]$$

where E_{100} is the emf E in steam at temperature t_s (100 °C) and E_0 is the emf with the working junction of the thermocouple in ice (0 °C). For this data set, the values of E_0 and E_{100} are 0.68 mV and 4.07 mV, respectively.

In the experiment, the thermo-emf of the Cu/Fe thermocouple, E_{CF}, is measured for values between the ice point and 450 °C. At the same time, the emf of the chromel/alumel thermocouple, E, is also recorded. In your DATA directory you will find a file, H15.DAT, which stores a set of data points in a three-columned "E E_{CF} $\sigma_{E_{CF}}$" format. We assume there is no uncertainty in the E values.

Using equation [1] the E data can be transformed into temperatures t. Theoretically E_{CF} and t are predicted to be related by,

$$E_{CF} = bt + ct^2. \qquad [2]$$

This is the equation of a parabola. It could alternatively be written as

$$E_{CF}/t = b + ct. \qquad [3]$$

Thus, a linear fit of E_{CF}/t versus t will give you the values for b and c.

Linear Least Squares Fit Equations

A **linear fit** to the equation $y = a_1x+a_0$ for a set of N data points with x and y values, and an uncertainty s_i associated with each measurement y_i, can be derived from the following equations:

$$a_1 = \frac{SS_{xy} - S_x S_y}{\Delta} \qquad a_0 = \frac{S_{xx}S_y - S_x S_{xy}}{\Delta}$$

with

$$\sigma_{a_1} = \sqrt{S/\Delta} \qquad \sigma_{a_0} = \sqrt{S_{xx}/\Delta}$$

where [4]

$$S \equiv \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \qquad S_x \equiv \sum_{i=1}^{N} \frac{x_i}{\sigma_i^2} \qquad S_y \equiv \sum_{i=1}^{N} \frac{y_i}{\sigma_i^2}$$

$$S_{xx} \equiv \sum_{i=1}^{N} \frac{x_i^2}{\sigma_i^2} \qquad S_{xy} \equiv \sum_{i=1}^{N} \frac{x_i y_i}{\sigma_i^2}$$

$$\Delta \equiv SS_{xx} - (S_x)^2$$

[from *Numerical Recipes: The Art of Scientific Computing*].

The study of the *E-t* relationship forms an important part of the study of the electrical properties of matter

The Program

Your task is to write a program to fit to the data to extract the least squares fit values for the parameters *b* and *c* in equations[2] and [3]. Then use these parameters to trace out the fitted curve (equation [2]) and plot on the same graph the original experimental values for E_{CF}, and *t*.

Don't forget to add useful comments

The program will need the following stages

[A] Input the data (E, E_{CF}, $\sigma_{E_{CF}}$) from "A:\DATA\H15.DAT" into arrays and determine the number of data points, *N*; print the value *N* on the screen with a delay afterwards; remember to dimension all of your arrays!

[B] Convert *E* to *t* (°C) using equation [1]: note that since there is no significant uncertainty in *E* that there will be none in *t*;

[C] Calculate E_{CF}/t and put the data with uncertainties into an array;

[D] Using a **subroutine procedure**, fit a line using the equations [4] to E_{CF}/t versus *t*, and hence determine *b* and *c* of equation [3] and their uncertainties;

[E] Using your estimates for b and c, their uncertainties, and equation [2], calculate a set of E_{CF} values and their uncertainties at the original t values. Output this data set (E_{CF} calculated, $\sigma_{E_{CF}}$ and t) to a *new* data file in the subdirectory A:\DATA\;

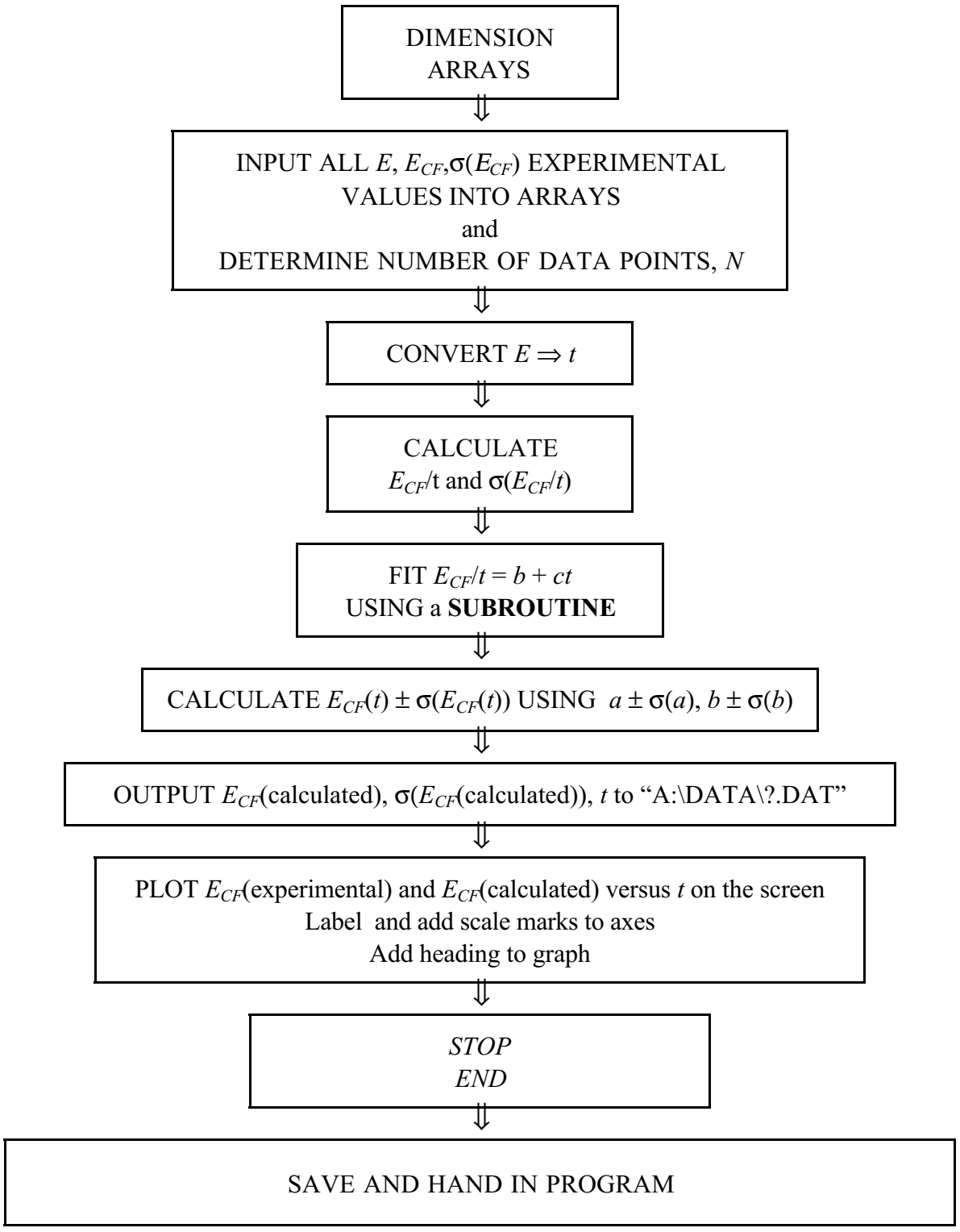
[F] On the screen, plot the original data and your calculated fit to the data. You need not plot the uncertainties on all of the individual experimental points just at a few representative points across the range.

[G] Save your program as "A:\PROGRAMS\WEEK7.BAS" and hand it in ....

These steps are summarised in the flow chart on the next page.

You should start this final task in week 6 or 7 of the QBasic course when I will be able to help you get started. You will then have the remainder of the second term in which to finish it in your own time, the deadline for handing in the work being the final Thursday of the term. I will be happy to help you with difficulties on an individual basis if you come and see me. Do not put the work aside until near the end of term, keep working on it and get it out of the way. Note that you will benefit greatly from sitting down and planning (and maybe even writing out!) your program *before* you try to type it in. In any case ... good luck

Flow Chart for A:\PROGRAMS\WEEK7.BAS



APPENDIX D

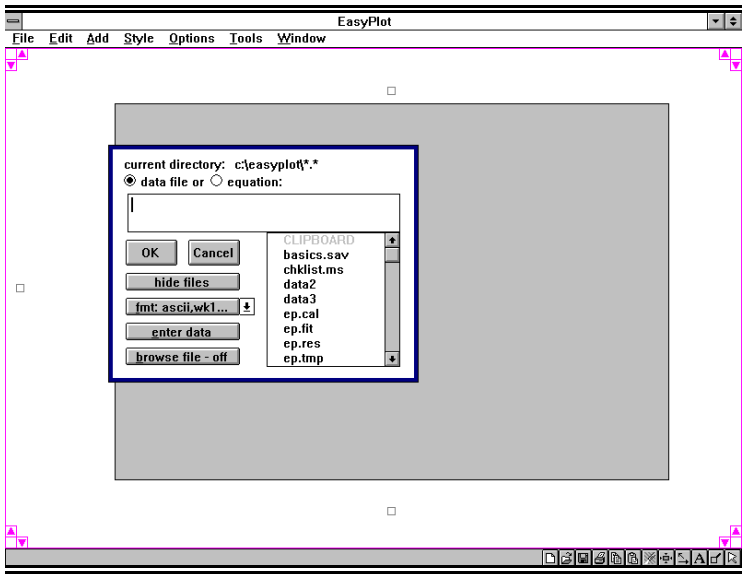
LAB DATA ANALYSIS PROGRAMS

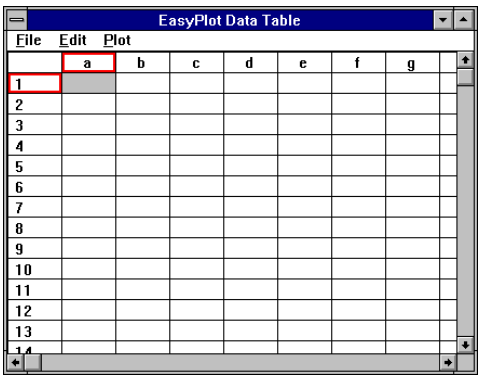
1. LSFIT

Easyplot is used to least squares fit a straight line to your data and plot the results. It calculates the best fit parameters for the slope and intercept but unfortunately does not give you estimates of the uncertainties on these parameters. To calculate these the data must be entered in the Lsfit program. Enter this by clicking on the Lsfit icon. The program instructs you as to how to proceed via explicit menus which come up on the screen.

2. EasyPlot

To start EasyPlot, double-click on the EasyPlot icon in the EasyPlot group window. You should then be presented with the EasyPlot screen, as below, with a dialogue box in the middle, asking you for a name of a data file to plot.





Rather than open a data file, select the **enter data** option, by moving the cursor over it with the mouse and clicking once with the left mouse button. You will then be presented with the EasyPlot Data Table window, as left, in which you can enter data. You can enter numerical data in rows and columns, but if you enter text, EasyPlot will treat the line the text is on as a whole, with no columns, and ignore any data entered there.

Until you know more about

EasyPlot's abilities, you will use the first lines of your data to tell EasyPlot exactly what you want to plot, and how you want your data interpreted. In this example, you will give EasyPlot a set of x- and y-values, with errors on the y-values. EasyPlot can handle errors in two ways; either as percentages of the plotted value (EasyPlot does this as default), or as actual, or absolute values. In this case the error values are absolute, so you need to tell EasyPlot this. In the first line, type

//abs_err (then press the return key) (the _ is an underscore, not a hyphen)

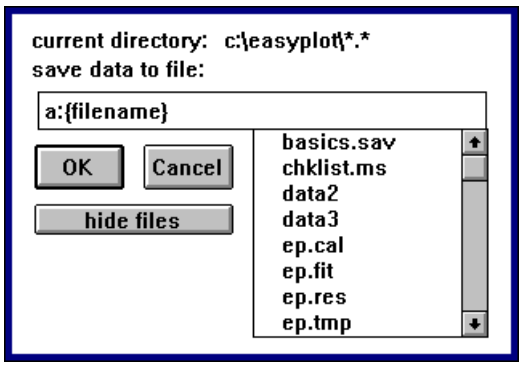
This is a batch command (more about these later), and tells EasyPlot that any error data following is to be treated as absolute values. The error bars plotted will have the y-value at their centre. (If you want the error bar to be plotted either side of the y-value, use the command //**half_err** instead)

Next EasyPlot needs to know what sets of data you are going to give it, for example, x- and y-values, or one x- and two sets of y-values; with errors on some, all or none of them. This is done using another batch command, thus:

/td xye (return) (the xye tells EasyPlot there is one column of x-values, one of y-values, and one of errors (for the preceding column, in this case y-errors)

Now type in the following values, in separate boxes, or **cells**, in the data table, below the two text lines. Use either the mouse or cursor (arrow) keys to move from cell to cell. If you enter a wrong value, use the backspace or delete key to edit it.

0	0.1	0.2
1	0.9	0.2
2	2.1	0.2
3	3.3	0.2
4	3.3	2.0
5	3.3	2.0



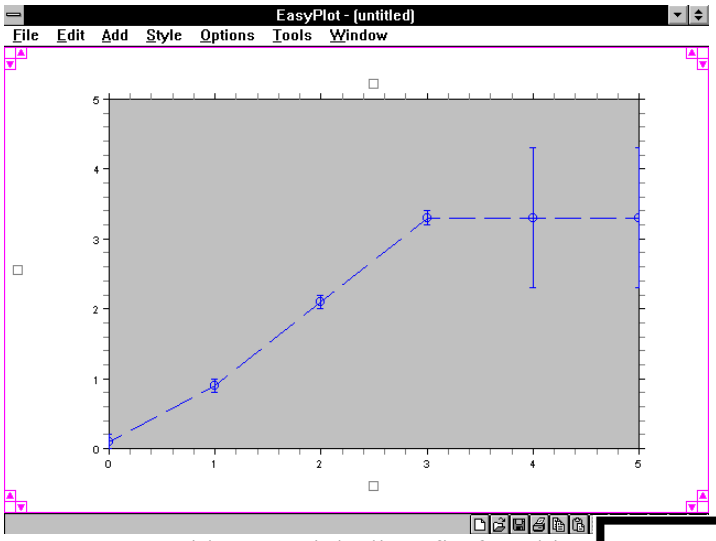
Before you save this data file, check that the top line of your data table has the symbols **x** (for x-values), **y** (for y-values) and **e** (for errors) in the three first columns. The **e** symbol will probably be missing in column c, so place the cursor over the wrong symbol and left-click. A menu with 'x', 'y', 'e' etc should appear. Left-click on 'e' so that EasyPlot will interpret the third column of data as error values.

Now save your data file. To do this, choose the **File** menu in the EasyPlot Data Table window, and select **Save As**. You will be presented with a screen similar to the one on the right. In the window below the words '`Save data to file:`', type a: (for the a: drive, containing your floppy disk), followed by **filename**; your choice of name for the data file. This must be no longer than eight characters long, and contain no spaces or punctuation marks. Click on the OK button to save your work.

Close the Data Table window by choosing the **File** menu at the top left of the Data Table window (do this by clicking the left mouse button when the cursor is over the word File), and then selecting **Exit**. This should leave you in the EasyPlot main screen.

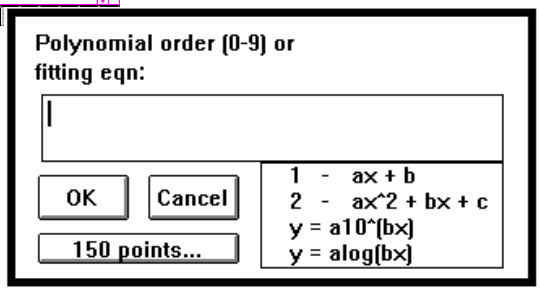
Before you plot your graph, you need to tell EasyPlot to draw the error bars. Select the **Style** menu, **Error Bars**, and then Y-axis, for error bars on the Y-axis. If there is a tick on the Y-axis line, do not select it again, as this will turn the error bars off.

Plotting your work



Now plot the graph. Do this by selecting the **File** menu, then click on **Open.** You will see the same dialogue box as the one at the start of the script. Instead of clicking on the 'enter data' button, type in your filename and then click on the OK button. You should now see a graph with error bars like the one shown above.

Adding a Straight-line fit



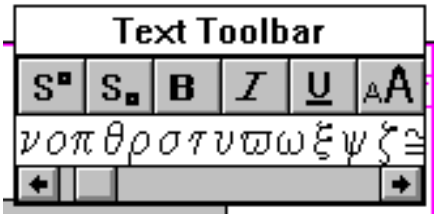
To provide a straight-line fit for this graph, choose the **Tools** menu. Select **Curve Fit**. You will see a dialogue box asking you for the number of the polynomial order to try and fit, or for a fitting equation. (see right).

Click on 1 to get the straight line fit y=ax+b and then click on the OK button. The screen will print out the fitted version of the line (values for a & b) and then it will ask you for the name of the file in which to save the resultant line fit data. Unless you have produced a very complicated graph, it is probably better not to save the graph, but rather to reconstruct it from your original table of data (which you should have saved to the floppy already).

Note that Easyplot provides printed values of the slope (a) and the intercept (b) next to the graph. Unfortunately, Easyplot does not calculate the *uncertainty* on the slope and intercept pf a graph (you will learn about uncertainty analysis as part of the data analysis course later in term 1). To find the uncertainties you must use a different program called lsfit (least squares fit) which will be described in the next section.

Titling your Work.

Adding a title, or labels for the x and y axes is quite easy. You should see three small squares at the top, left and bottom of the plot. Left-clicking on any one of these produces a text window, in which you can type an x-label, a y-label or a title as required. When entering any form of text in EasyPlot, a **Text Toolbar** window appears in the upper right-hand part of the screen. This has several icons along its top representing functions which are, from left to right:



Superscript Enables you to enter text as superscript, e.g. X^2.

Subscript	Enables you to enter text as subscript, e.g. $X_{\mathbf{0}}$.

To get out of superscript or subscript, move the cursor to the right, using the right arrow key.

Bold	Enables you to embolden text, e.g. **Bold**.
Italics	Enables you to italicise text, e.g. *Italics*.
Underline	Enables you to underline text, e.g. Underlined.
Text size	Enables you to change the size of your text, e.g. small medium large extra large

You can also enter symbols or Greek characters by selecting the symbol that you require from the choice below the icons. Use the arrows at either side below the symbol window to show more symbols.

Printing your graph

To print out your graph, choose the File menu, then Print. When asked where to send the image to, press p (for printer) and click on the OK button.

This very simple introduction to EasyPlot will enable you to perform most of the plotting you will need during the year. However, the program is actually much more complex and useful than this, and you may subsequently wish to utilise this complexity. To do so you will find extra instructions for EasyPlot provided around the laboratory.