Part (c) Program Development

(i) Implementing the program

The problem has been identified and the solution has been designed. Candidates must now write original code that will solve the problem and be related to the design that they plan to use. There are plenty of other marks available in the rest of section (c) for more specific areas of the programming. The marks here are for producing a solution to the problem that needs to be solved. Candidates need to produce a program listing, probably 2 sides of A4, any more and the problem and/or the solution is too complex. There will be a printout of the data structure (file) that has been produced, along with printouts of the input and output screens, probably another 2 sides of A4. The program listing should be clearly annotated (by hand is fine) to show how the code relates to the design. Notice that there is no real need to print out further copies of the code as evidence for c (i) and c (ii) because the evidence for these sections can be shown on this printout.

Evidence required

- program code which is annotated by hand to show how it relates to the design section
- printout to show the structure (probably with the data included) of the file that has been created
- printout to show input screens and output screens (perhaps a number on a single page). These outputs should be hand annotated to explain their purpose and under what circumstances they will be used.

(ii) Using good Programming Style

This section of the project is partly common sense and partly designed to provide practical experience of the techniques in section 1.3 (h). The code should be identifiable, so it should contain the details about the author, the centre and the date and version. The code should also be described, so it should contain details of the programming language used and the purpose for which the code was produced. This is not meant to be difficult for the candidate to do, but it is intended to encourage good practice. Note that there may be occasions when it is not necessary to produce so much detail, while on other occasions the candidate may decide that further detail is necessary in the header, perhaps listing the titles of procedures and library routines that are being accessed. In addition to the information in the header, candidates are expected to make the code self documenting. There is a limit to the value of this as the code is not expected to be very lengthy, but the candidate is expected to ensure that all the normal techniques are illustrated as part of their submission.

It is necessary to understand that the purpose of this project is to identify the skills of the candidate, not the ability to printout code which is automatically generated by generic software packages. It is a good idea not to allow use of generic software, then this problem does not arise. However, if it is essential, then any code that has not been produced by the candidate but is essential to the solution to the problem must be clearly identified.

Evidence required

- the code must have a recognised and accurate header containing appropriate detail. It is strongly suggested that, unless there is a good reason to the contrary, the list in the syllabus should be used
- the code should illustrate the following techniques
  - data names, procedure and function names should all be sensible and relate to the data being stored
  - data, procedures and functions should be explained when they are declared
  - procedures and functions should be clearly delineated from each other and from the main program
  - indentation should be clearly used to distinguish selection and loop constructs clearly
  - comments should be used widely to explain meanings in the code

  in order to make the code understandable to a third party observer

(iii) Programming skills

There are eight listed skills that the candidate has to demonstrate. These should arise naturally from the requirements of the problem if the problem has been chosen sensibly.

Each of the eight skills should be evident in the code listing and should be clearly identified, either by use of annotation within the code or by written annotation. The method is not really important, save for two things. The first is that the moderator must be able to identify the eight techniques, so make it clear! And secondly, the use of annotation within the code needs to be illustrated for (ii) so it might as well be used here.

Evidence required

For each of the eight techniques

- a valid use within the context of the problem being solved
- sensibly and clearly annotated so that the techniques can be clearly identified within the code.

Notice from the evidence above that it is expected that the technique should arise from within the context of the problem being solved.

Consider the problem of the mark book for the teacher which is being used throughout this section as the example problem to be solved.

1. Arrays and/or records. The mark book is going to consist of a record for each &8 student in the maths set. The student is going to create a file structure using the relevant command within the language used and will then proceed to fill the structure with the data which will consist of individual records. A 2 dimensional

array can be used to read the file into the memory so that it can then be used in the processing. Note that all these techniques, and some of the others, tend to overlap. Do not worry if you find that you are using the same piece of evidence for more than one of the techniques, though it would be better if they could be done separately.

2. The teacher needs to store and use the students' names (text), their marks (integers), a mean mark (real), an indication of students who require extra lessons (Boolean). The list is only really limited by the imagination, but each one needs to be specifically related to the problem being solved.
3. The teacher wishes to identify all the students who have scored less than half marks for a homework so that they can be given extra practice. A simple statement of the form IF X < 50 THEN… is all that is needed, but make sure that it is clearly flagged and that it arises naturally from the problem being solved.
4. The selection procedure in number 3 is, presumably, going to be carried out on all the students in the set, so a process will be set up to look at each of the students in turn (a simple loop construct seems sensible), this provides the iteration.
5. The opening screen when the program is run is a simple menu which asks the teacher which of a number of options (e.g. add a new student, work out the mean marks, select students whose have scored less than half marks, add a new mark for each student) is wanted. Each of the options leads to a different procedure. It is possible to include the use of a parameter if the student particularly wants to, but why bother?
6. One of the options is to calculate the mean mark for a piece of work. The code leads to a procedure which reads the marks for the piece of work and then divides by the number of marks that have been added together. The answer is then available to the rest of the code as the mean mark. This is a procedure which returns a single value and is consequently a function.
7. This has already been carried out in 3 and 4. However, for clarity we will imagine that the teacher can search for a specific student and print out that student's scores. This is slightly different because having found the student, the search is completed without necessarily reading the whole file.