OCR
RECOGNISING ACHIEVEMENT

**OXFORD CAMBRIDGE AND RSA EXAMINATIONS**

**General Certificate of Education Advanced Subsidiary Level**

# COMPUTING 2507

Structured Practical Computing Tasks

Issued September 2005
Maximum mark **120**

**JUNE 2006**

**OPEN ON RECEIPT**

**INSTRUCTIONS TO CANDIDATES**

- You should attempt all tasks, working independently from other candidates.
- There are no time limitations on the tasks other than that they must be submitted by the appropriate internal deadline set by the Candidate's Centre. This deadline will reflect the need for the Centre to complete marking of the tasks and submit the marks to OCR by the required date.
- There are no restrictions on computing facilities, hardware or software, that may be used.
- You are strongly advised to keep all your working notes as these may be required by the moderator.
- Reasons for answers to tasks are expected to form part of the work submitted.

**Notice to candidates**

1 The work which you submit for assessment must be your own.
However, you may:
**(a)** quote from books or any other sources: if you do, you must state which ones you have used;
**(b)** receive guidance from someone other than your teacher: if so, you must tell your teacher, who will record the nature of the assistance given to you.

2 If you copy from someone else or allow another candidate to copy from you, or if you cheat in any other way, **you may be disqualified from at least the subject concerned**.

3 When you hand in your coursework for assessment, you will be required to sign that you have understood and followed the coursework and portfolio requirements for the subject.

**ALWAYS REMEMBER – YOUR WORK MUST BE YOUR OWN**

**This question paper consists of 7 printed pages and 1 blank page.**

**[Turn over**

**Task 1 [40 marks]**

**This is a software development task and an implementation task.**

A film society has over 300 members who are able to borrow from its library of more than 1000 classic films. There is more than one copy of some films. The films may be held in videocassette form, DVD form, or, in some cases, both.

Members pay an annual membership fee that allows them to borrow films free of charge. Members can only have one film out on loan at a time. Loans are not permitted to extend beyond seven days.

The film society needs a database to contain details of its members, its films and those films currently out on loan.

The table of members is to contain appropriate contact details for each member and the date on which their membership fee next becomes due. It should also ensure that a member can only borrow one film at a time.

The table of films is to contain information about each film copy. It should store the name of the film, its director, its year of release and running time, as well as a brief synopsis of the storyline.

The loans table will identify which films are out on loan and to whom.

**(a)** Create a table called Member to hold details of members of the film society.

Explain the reason for including each attribute (field). Give the data type of each attribute. Identify the key. [6]

**(b)** Create a table called Film to hold details of films owned by the film society.

Explain the reason for including each attribute (field). Give the data type of each attribute. Identify the key. [5]

**(c)** Create a table called Loan to hold information about films out on loan at any one time.

Explain the reason for including each attribute (field). Give the data type of each attribute. Identify the key. [5]

**(d)** Create suitable data for each of your tables. Although the film society has over 300 members and more than 1000 films, your tables need only show a sample of the records the film society would hold. You should not attempt to create the whole database. Instead, your Member and Film tables should each hold at least 20 records. Your Loan table should hold at least 10 records. You should choose your data to be suitable for producing sensible results in the remaining questions.

Include a full printout of each of your tables. Screen shots are acceptable. [8]

At the film society library, members should be able to interrogate the database. They should be able to

- print a hard copy report of information about an individual film title.
- print a hard copy report listing all films made by an individual director.
- run an alphabetical screen listing of all films currently available for loan.

**(e)** Create interfaces (of which there should be hard copy evidence) that allow a member to

**(i)** input the name of a film and output to the printer the information held about that film.

Provide at least **two** sample reports.

**(ii)** input the name of a director and output to the printer a list of the society's films made by that director (with the years of release and running times).

Provide at least **two** sample reports.

**(iii)** output to the screen a list, in alphabetical order, of all films currently available for loan.

Provide hard copy of **one** sample page of your screen.

[9]

**(f)** Create an interface that allows the film society to send mail merge letters to any members whose loans are shown to be overdue. Provide hard copy evidence of your standard letter (with mail merge fields) and show that your mail merge produces suitable letters to all members whose borrowed films are shown as overdue in the Loan table. [7]

**[Turn over**

**Task 2 [16 Marks]**

**This is an algorithm trace task. No implementation is required.**

Read the following algorithm, which operates on numbers held in an array **Sift**, where the **size** of the array is 10. All of the variables used in the algorithm are integers.

In the algorithm, **DIV** is an operator that divides one integer by a second integer and truncates the result to its integer part. This means that

15 **DIV** 4 is 3
19 **DIV** 2 is 9

```
0    size = 10
1    sep = size DIV 2
2    WHILE sep > 0
3        FOR count = 1 TO size - sep
4            index = count
5            WHILE (index > 0) AND (Sift[index] > Sift[index + sep])
6                spare = Sift[index]
7                Sift[index] = Sift[index + sep]
8                Sift[index + sep] = spare
9                index = index - sep
10           ENDWHILE
11       NEXT count
12       sep = sep DIV 2
13   ENDWHILE
```

Set up a table with headings and initial values as shown below. You are required to build a table that will show all the changes to variables and the cells of the array as the algorithm is traced from start to finish. You should only show a value in the Variables columns when the algorithm writes a value to that cell. You should, however, complete **all** of the cells of the array Sift each time **any** cell in the array Sift is written to. Develop the table by working from left to right and top to bottom. This means that when you need to write a new value in a cell that is to the left of the last cell you wrote in, you must move to the next row.

| Variables | | | | | Array Sift | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| sep | count | index | spare | size | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 0 | 0 | 0 | 10 | 103 | 64 | 18 | 7 | 144 | 83 | 15 | 37 | 11 | 52 |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |

[16]

**Task 3 [41 marks]**

**This is a software development task and an implementation task.**

Roman numerals continue to be used for certain limited purposes such as copyright dates. The symbols used are I=1, V=5, X=10, L=50, C=100, D=500 and M=1000.

One definition of Roman numbers is

- the letters must be in descending order of value,
- the value of the number is the sum of all the letters,
- a letter must not be repeated if it can be replaced by another letter.

This is known as the additive rule and is the only rule to be used in this question.
Note that Roman numbers are always integers and cannot have fractional values.

For example

❑ MCX = 1000 + 100 + 10 = 1110
❑ MXC is invalid because of wrong order
❑ IIII = 1 + 1 + 1 + 1 = 4
❑ IIIII is invalid because it can be replaced with V

A manufacturing company wishes to make a new calculator that allows a user to input two numbers using Roman numerals and displays the numbers in both Roman and Arabic decimal forms. The calculator then allows a user to add or subtract the two numbers and again displays the result in both Roman and Arabic decimal forms.

The calculator must

- allow a user to enter two numbers in Roman numerals, according to the additive rule, each up to a maximum value of MMMM (4000),
- prevent the user entering an invalid Roman numeral,
- display the numbers entered in both Roman numerals and in their Arabic decimal equivalent,
- allow the user to enter an operator for addition or subtraction,
- prevent the result of a subtraction being zero or a negative number,
- display the result of the calculation in both Roman and Arabic decimal forms,
- allow the user to clear all displays.

[Turn over

The company wishes to use a simulation of this calculator so that the user interface can be tested.

You should use a high level language to complete the following tasks. State the language and version you use.

**(a)** Design and implement a user interface that can be used to test the above facilities.

Provide hard copy evidence of your design. [8]

**(b)** Write a function that has, as a parameter, an already validated Roman number and returns its Arabic decimal equivalent.

Provide hard copy of your function code, which should be fully annotated.

Write a short program to test your function. Use a Roman number that contains at least three different Roman numerals.

Provide hard copy evidence of your test. [5]

**(c)** Write a function that has, as a parameter, an Arabic decimal number and returns its Roman number equivalent.

Provide hard copy of your function code, which should be fully annotated.

Write a short program to test your function. Use an Arabic decimal number that converts to a Roman number containing at least three different Roman numerals.

Provide hard copy evidence of your test. [5]

**(d)** Write a program that will enable your simulated calculator to add and subtract two Roman numbers. Your program must use the interface and the two functions you have created in **(a)**, **(b)** and **(c)**.

Provide hard copy of your program code, which should be fully annotated. [11]

**(e)** Provide a table to show data suitable for testing the program you have written in **(d)**. The table should show at least six different tests. In each case you should indicate the reason for the test, the data to be used and the expected result.

Provide hard copy evidence showing the use of your test data. Any error messages should appear on the calculator's interface. [12]

**Task 4 [23 marks]**

**This is an algorithm trace task. No implementation is required.**

Read the following algorithm for the recursive function called Curiosity that needs two integers as inputs and returns an integer value.

In the algorithm, **MOD** is an operator. It works with two integers to give the remainder when the first integer is divided by the second. This means that

```
17 MOD  7 is  3
72 MOD 15 is 12
```

```
FUNCTION Curiosity(X:Integer;Y:Integer):Integer;
   IF Y = 0 THEN
      Curiosity = X
   ELSE
      Output Y and (X MOD Y)
      Curiosity = Curiosity(Y,X MOD Y)
   ENDIF
END Curiosity
```

**(a)** Write down the output when the function is called with

**(i)** Output 'The answer is ', Curiosity(120,42)

**(ii)** Output 'The answer is ', Curiosity(221,153)

[8]

**(b)** The function is called with

Output 'The answer is ', Curiosity(30,24)

Write down in full each line **that is executed**, but replace each variable with its actual value and indicate on the IF statement line whether the result of the test is true or false. [7]

**(c)** Describe the purpose of the algorithm. [2]

**(d)** Rewrite the recursive function as an iterative algorithm. [6]

BLANK PAGE