RECOGNISING ACHIEVEMENT

ADVANCED SUBSIDIARY GCE **2507**

COMPUTING

Structured Practical Computing Tasks

JUNE 2008 Issued September 2007

Maximum Mark **120**

OPEN ON RECEIPT

CUP/T53644

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 document consists of **11** printed pages and **1** blank page.

Task 1 [39 marks]

This is a software development task and an implementation task.

A university runs modular degrees. Each student takes examinations in many modules and many students study each module. In this task you are to create a simple database to record details of the students, modules and the students' results.

Students are sent their results by post but may need to be contacted by telephone or email. It is also necessary to include their date of birth and their ID, which is an 8-digit number and may start with a zero.

Modules have a unique identifier and a name. Each module is worth 15 or 30 credits.

A student takes module examinations in January and June each year and may be entered for an examination more than once, but not in the same examination session. The result of an examination is given as a percentage.

The database consists of three tables called STUDENT, MODULE and RESULT.

(a) Create the table STUDENT to hold details of the students.
- Give the reason for including each of the attributes.
- Give the data type of each attribute.
- Identify the key.
- Explain how you have created validation rules.

Provide hard copy of your design.
Screen shots are acceptable.

[10]

(b) Create the table MODULE to hold details of the modules.
- Give the reason for including each of the attributes.
- Give the data type of each attribute.
- Identify the key.
- Explain how you have created validation rules.

Provide hard copy of your design.
Screen shots are acceptable.

[4]

(c) Create the table RESULT to hold details of the students' results.
- Give the reason for including each of the attributes.
- Give the data type of each attribute.
- Identify the key.
- Explain how you have created validation rules.

Provide hard copy of your design.
Screen shots are acceptable.

[7]

(d) Create suitable data for each of your tables.
- Include at least 20 students and 10 modules.
- The RESULT table should have at least 30 entries.
- Choose your data to be suitable for producing sensible results in the remaining questions.

Provide a hard copy of the whole of each of your tables.
Screen shots are acceptable.

[8]

(e) Create a report that shows the female students' ID's and names and all the details of all the modules they have taken. The report should have a suitable heading and it should be ordered by student name.

Provide

- *a hard copy of your design (a screen shot is acceptable)*
- *a hard copy of your report (a screen shot is **not** acceptable).*

[5]

(f) Create a report that shows the details of all the students, with their results, for a given module, session and year. The module, session and year should be entered by the user when the report is requested. Students should be listed alphabetically.

Provide

- *a hard copy of your design (a screen shot is acceptable)*
- *a hard copy of your report (a screen shot is **not** acceptable).*

[5]

Task 2 [32 marks]

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

The following algorithm uses one procedure and five functions.

The procedure PUSH(anArray, top, ch) first adds one to the value of top and then stores the single character held in ch in the one-dimensional array anArray at anArray[top].

The function POP(anArray, top) returns the character held in anArray[top] and **then** takes 1 off the value of top.

The function NEXTCHAR(str) returns the next character in its argument str which is a string.

The functions F(ch), G(ch) and R(ch) take a single character as input and return a value according to the table in Fig. 2(a). For example

F(a single letter) = 7, G(a single letter) = 8 and R(a single letter) = 1

That is

F("a") = 7, G("b") = 8 and R("c") = 1

Character (ch)	**Function F(ch)**	**Function G(ch)**	**Function R(ch)**
+, -	1	2	-1
*, /	3	4	-1
^	6	5	-1
A single letter	7	8	1
(	9	0	-
)	0	-	-

Fig. 2(a)

In the algorithm, & joins two strings together (concatenation).
For example,"a + b" &")" = "a + b)"

```
1. INPUT aString
2. aString = aString & ")"
3. top = 0
4. PUSH(anArray, top, "(")
5. newString = ""
6. rank = 0
7. current = NEXTCHAR(aString)
8. WHILE current <> ""
9.   IF top < 1 THEN
10.     OUTPUT "Invalid string"
11.     END the procedure
12.   ENDIF
13.   WHILE F(current) < G(anArray[top])
14.     temp = POP(anArray, top)
15.     newString = newString & temp
16.     rank = rank + R(temp)
17.     IF rank < 1 THEN
18.          OUTPUT "Invalid string"
19.          END the procedure
20.     ENDIF
21.   ENDWHILE
22.   IF F(current) <> G(anArray[top]) THEN
23.     PUSH(anArray, top, current)
24.   ELSE
25.     POP(anArray, top)
26.   ENDIF
27.     current = NEXTCHAR(aString)
28. ENDWHILE
29. IF top <> 0 OR rank <> 1 THEN
30.   OUTPUT "Invalid string"
31. ELSE
32.   OUTPUT "Valid string"
33. ENDIF
34. END of procedure
```

Fig. 2(b) shows the values of current, anArray, newString and rank and the output when the input string is

"a+b"

current	**anArray**	**newString**	**rank**
	(		0
a	(a		
+	(	a	1
	(+		
b	(+b		
)	(+	ab	2
	(	ab+	1
	empty		
empty			
		Output is:	Valid string

Fig. 2(b)

Produce a similar table for each of the following input strings. In your table, you should work from left to right and at the end of a row you should move onto the next row. You do not need to write a value again if it is the same as in the previous row.

(a) "cd" **[6]**

(b) "p*(q" **[9]**

(c) "(v-w)*(x+y)" **[17]**

BLANK PAGE

Task 3 [32 marks]

This is a software development task and an implementation task.

Wari is a game played on a board that consists of 12 cups placed in two parallel rows of six cups and a large cup at each end. The game is for two players; each player has a row of six cups and one large cup for prisoners. Initially, there are four marbles in each small cup, as shown in Fig. 3(a).

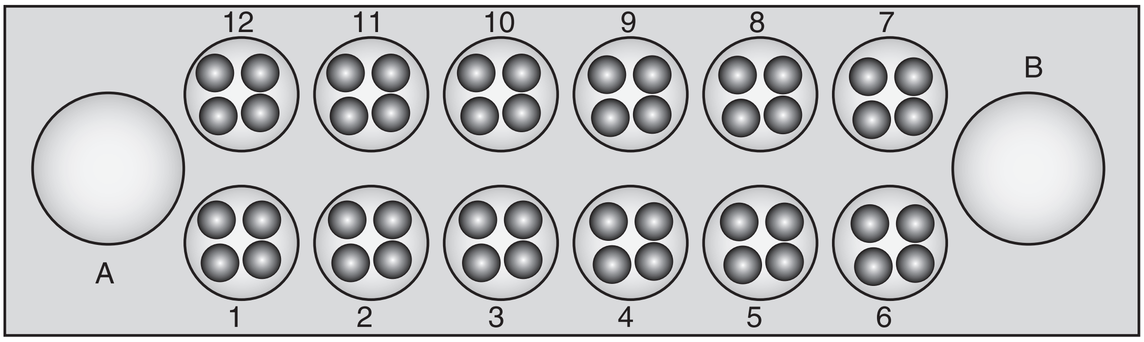

Fig. 3(a)

Player A uses cups 1 to 6 and player B uses cups 7 to 12.

The players decide who is to go first, and the first player takes all the marbles from one of his cups and places them one at a time into successive cups in an anticlockwise direction starting with the cup immediately to the right of the emptied cup and continues round his opponent's cups if necessary.

The second player then takes all the marbles from one of his cups and distributes them similarly in an anticlockwise direction.

Play continues until one of the players places his last marble in an opponent's cup that contains one or two marbles (i.e. two or three marbles after distribution). The player is entitled to all the marbles in this cup as prisoners. The player removes the marbles and places them in his large cup at the end of the board. If there are also two or three marbles in the preceding cup and this cup belongs to his opponent, the player is entitled to these also. This continues until a cup is encountered, on the opponent's side, that does not contain two or three marbles.

Now consider an example. Suppose player A goes first and removes the four marbles from cup 5 and distributes them into cups 6, 7, 8 and 9. Cup 9 does not contain two or three marbles so the turn ends. This can be represented by A(5,4).

Now suppose the following moves take place.

B(7,5), A(2,4) and B(12,5).

The position is now as shown in Fig. 3(b) and Fig. 3(c).

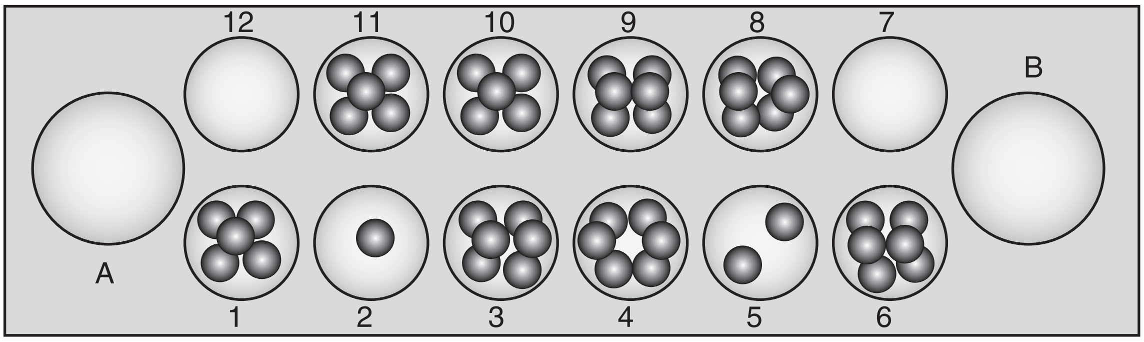

Fig. 3(b)

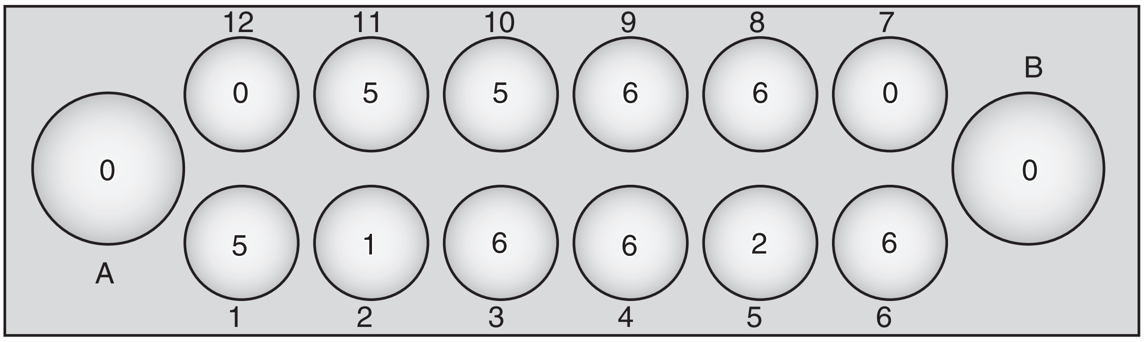

Fig. 3(c)

B's last play was into A's cup 5 which now has two marbles in it. These are now removed and are B's prisoners. B places these marbles in the cup labelled B.

If twelve or more marbles are in a cup and are distributed, the twelfth marble is placed into the cup after the one from which the marbles were removed. That is, the cup from which the marbles were removed remains empty.

A player may not empty or leave empty all the cups on the opponent's side. The player must play to leave at least one marble on the opponent's side. If this is not possible, the player removes all the marbles from his side of the board and adds them to his prisoners and the game is over. The winner is the player with the most prisoners.

Sometimes it happens that no player can win. For example, player A has a marble in cup 3 and player B has a marble in cup 9; in this case the marbles simply follow one another round the board. In this case the players remove the marbles and add them to their prisoners and the game is over.

You are advised to play the example game using the rules given.

You are to write a program that will decide the winner (if any) of a game of Wari.

To do this, you must use a high level language. **State the language and the version you use.**

You are expected to provide full annotation of your program code and to use meaningful names throughout.

You must complete the following tasks, **each of which must be presented separately,** in your final submission.

(a) Write annotated code that will create data structures that will represent the board and initialise their contents.

You should include a listing of this code. **[6]**

(b) Design and implement an interface that will show the players the number of marbles in each cup, including the large cups. You should use the data structures from (a).

A screen shot of the interface is acceptable.

You should include a listing of this code. **[5]**

(c) Write annotated code that will get the identity of the player that is to go first or allow the game to be terminated.

You should include a listing of this code and any testing that you have done. **[4]**

(d) Write annotated code that will get the identity of the cup that is to be emptied or an indication that the player wishes to surrender. The code should fully validate the input.

You should include a listing of this code and any testing that you have done. **[5]**

(e) Write annotated code that will distribute the marbles when a move has been made. This code should identify and empty the cups according to the rules. The number of marbles in each cup, including the large ones, after the move is completed should be displayed ready for the next player.

You should include a listing of this code and any testing that you have done. **[5]**

(f) Write annotated code that allows two players to play a game of Wari. The game should continue automatically until a winner is found or a player surrenders.

You should include a listing of your complete code. **[7]**

Task 4 [17 marks]

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

Read the following algorithm for the recursive procedure called Mystery.

```
MYSTERY(soFar, toGo)
  length = LEN(toGo)
  IF length = 1 THEN
   PRINT soFar & toGo
  ELSE
   FOR i = 1 to length
     MYSTERY(soFar & MID$(toGo, i, 1),LEFT$(toGo, i - 1)
                        & RIGHT$(toGo, length - i))
   NEXT i
  ENDIF
END of procedure
```

Where

`LEN(aSTRING)` returns the number of characters in the string held in aString,
`MID$(aString, n,m)` returns m characters of aString starting at the nth,
`LEFT$(aString, n)` returns the first n characters of aString,
`RIGHT$(aString, n)` returns the last n characters of aString,

`&` joins two strings together (concatenation).

(a) Give the output if the procedure is called by

MYSTERY("", "RED")

[7]

(b) The function is called with

MYSTERY("", "AB")

Write down all the instructions, including procedure calls, in the order in which they are executed. You must show the values of the variables in each instruction. **[10]**