OCR
Oxford Cambridge and RSA

A Level Computer Science

H446/02 Algorithms and programming

Practice paper - Set 1

Time allowed: 2 hours 30 minutes

Do not use:
- a calculator

First name									
Last name									
Centre number						**Candidate number**				

INSTRUCTIONS
- Use black ink.
- Complete the boxes above with your name, centre number and candidate number.
- Answer **all** the questions.
- Write your answer to each question in the space provided.
- If additional space is required, use the lined page(s) at the end of this booklet. The question number(s) must be clearly shown.
- Do **not** write in the barcodes.

INFORMATION
- The total mark for this paper is **140**.
- The marks for each question are shown in brackets **[ ]**.
- Quality of extended responses will be assessed in questions marked with an asterisk (*).
- This document consists of **28** pages.

Answer **all** the questions.

Section A

1 A flight simulator allows a user to take control of a simulated aeroplane. The user can fly the plane in an environment that can simulate different weather conditions and additional planes in the sky.

(a) Identify **three** pieces of information that would need to be researched in order to design this simulator.

1. .....

.....

2. .....

.....

3. .....

..... **[3]**

(b) Explain what is meant by 'concurrent processing' and describe **one** example of how the simulator could make use of it.

Concurrent processing .....

.....

Example .....

.....

.....

.....

.....

..... **[4]**

(c) Air traffic controllers are considering introducing a new flight path.

Explain **two** reasons why they might use the new flight path in the simulation before implementing it in the real world.

1. .....

.....

.....

2. .....

.....

.....

..... **[4]**

(d) Abstraction has been used in the design and creation of the flight simulator.

Explain, using an example, the need for abstraction in the creation of the flight simulator.

.....

.....

.....

.....

.....

..... **[3]**

2 The layout for a 2-player board game is shown in Fig 2.1

START	1	2	3	4	5	6	7
15	14	13	12	11	10	9	8
16	17	18	19	20	21	22	23
31	30	29	28	27	26	25	24
32	33	34	35	36	37	38	39
47	46	45	44	43	42	41	40
48	49	50	51	52	53	54	55
END	62	61	60	59	58	57	56

Fig 2.1

The game is played by rolling two 6-sided dice and moving that number of spaces. Both players start on the START space. If a player lands on a space occupied by the other player, they move to the next available space.

The board is to be stored as a 2-dimensional array.

(a) The board shown in Fig 2.1 is a visualisation of the problem. Explain what visualisation means in this example.

.....

.....

.....

..... **[2]**

(b) Each time a player moves, a series of obstacles are to be added to the board.

On their turn, each player rolls two dice. The smaller number from the two dice is taken, and that many obstacles will appear on the board in random locations.

For example, if a 3 and 6 are rolled, then 3 obstacles will appear.

A recursive function is written in pseudocode to perform this task.

```
01 function generateObstacle(diceNumber)
02     if diceNumber == 0 then
03         return true
04     else
05         x = randomNumber(0, 7)
06         y = randomNumber(0, 7)
07         board(x, y) = new obstacle()
08         generateObstacle(diceNumber-1)
09     endif
10 endfunction
```

The code `new obstacle()` generates an instance of the object obstacle.

(i) Explain the purpose of the code in line `01` in the algorithm.

.....

.....

.....

..... **[2]**

(ii) Identify the line of code where recursion occurs.

..... **[1]**

(iii) The recursive function could have been written using iteration.

Describe the benefits and drawbacks of using recursion instead of iteration.

Benefits .....

.....

.....

.....

Drawbacks .....

.....

.....

..... **[4]**

(iv) Rewrite the function in part (b) so it uses iteration instead of recursion.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... **[4]**

(v) If a position on the board is not occupied, its value is set to a blank string ("").

The current algorithm does not check if the random space generated is currently occupied.

Write a subroutine that takes the generated position of the board, checks if it is free and returns `true` if free, or `false` if occupied.

.....

.....

.....

.....

.....

.....

.....

.....

..... **[3]**

(c) The programmer is using a number of subroutines in the program. Explain, using an example, the benefits to the programmer of using subroutines in the creation of this game.

.....

.....

.....

.....

.....

.....

.....

.....

..... **[4]**

(d) The programmer has been told the recursive function has the Big O notation of O(n).

(i) State the purpose of Big O notation.

.....

..... **[1]**

(ii) Explain what the Big O notation O(n) means for this recursive function.

.....

..... **[1]**

3* A multi-national retailer has a very large database storing customers, stock and orders.

The retailer uses data mining to retrieve a variety of information.

Evaluate the use of data mining by the retailer.

.... **[9]**

4 A 1-dimensionsal array stores a set of numbered cards from 0 to 7. An example of this data is shown in Fig in 4.1

2	0	1	7	4	3	5	6

Fig 4.1

(a) The programmer wants to search for a specific card in the array.

State whether a binary search or a linear search would be the most appropriate method to search for a specific card, and justify your answer.

Search method.....

Justification .....

.....

.....

..... **[3]**

(b) A programmer is writing a computer program to sort the cards into the correct order (0 to 7).

(i) Show how an insertion sort would sort the array in Fig 4.1 into the correct order. Draw the array after each move.

.....

.....

.....

.....

.....

.....

.....

.....

..... **[3]**

(ii) Describe how a quick sort algorithm works with the data in Fig 4.2.

2	0	1	7	4	3	5	6

Fig 4.2

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... **[6]**

(c)* Two other sorting algorithms the programmer could have used are a merge sort and bubble sort.

The worst case scenario for Merge is O(n log(n)) and for Bubble is O(n^2).

Compare the use of a merge sort and a bubble sort on this array, evaluating the performance of each sort, making reference to the worse case Big O notation.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... **[9]**

5 A procedure is shown below.

```
01   procedure fun1(x)
02       y=""
03       if x < 0 then
04           flag = true
05           x = x * -1
06       endif
07       while (x > 0)
08           y = str(x MOD 2) + y
09           x = x DIV 2
10       endwhile
11       if flag == true then
12           y = "1" + y
13       else
14           y = "0" + y
15       endif
16       print(y)
17   endprocedure
```

`flag` is a local variable and has a default value of `false`.

(a) Explain why `str` is needed in line `08`.

.....

.....

.....

.....

..... **[3]**

(b) (i) Show the result of `y` when the procedure is called with: `fun1(10)`. Show your working.

`y`: ..... **[4]**

(ii) Show the result of `y` when the procedure is called with `fun1(-13)`. Show your working.

`y`: ..... **[4]**

(b) (iii) Identify the purpose of this algorithm.

.....

..... **[1]**

(c) In this procedure, `flag` is assumed to be a local variable.

(i) Explain the problem that would be caused in this algorithm if `flag` was a global variable.

.....

.....

.....

.....

.....

..... **[3]**

(ii) The programmer has chosen to keep `flag` as a global variable.

Describe how the algorithm could be changed to prevent the error identified in part (i)

.....

..... **[1]**

6 A salesman travels around the country, stopping at specific places, and then returning to the starting place.

Fig 6.1 shows an example map of places that the salesman visits.

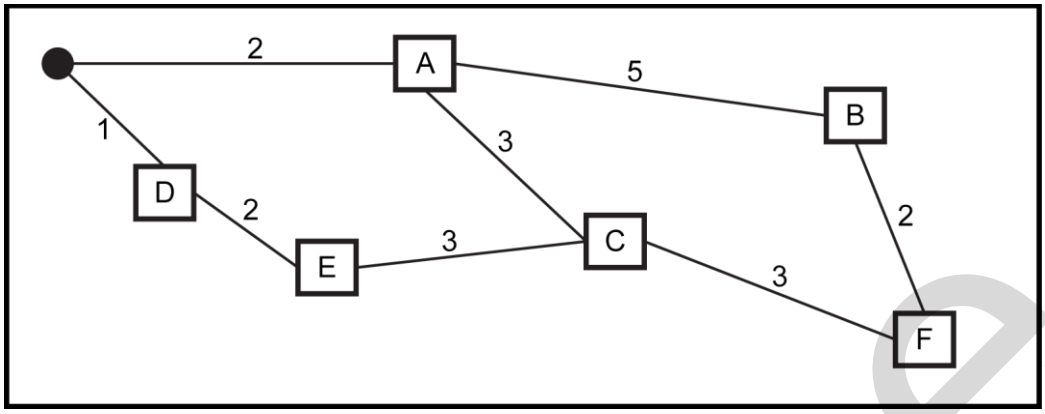

Fig 6.1

The filled in circle represents the start and end point. The letters represent the places to visit. The lines are the routes available and the numbers are the length of time each route takes to travel.

(a) Explain how abstraction has been applied in the production of Fig 6.1

.....

.....

.....

..... **[2]**

(b) The travelling salesman aims to find the shortest route between these places to visit.

A programmer is writing an algorithm to solve the travelling salesman problem.

The programmer is using a tree to find the most efficient route. Fig 6.2 shows part of the tree with three levels completed.

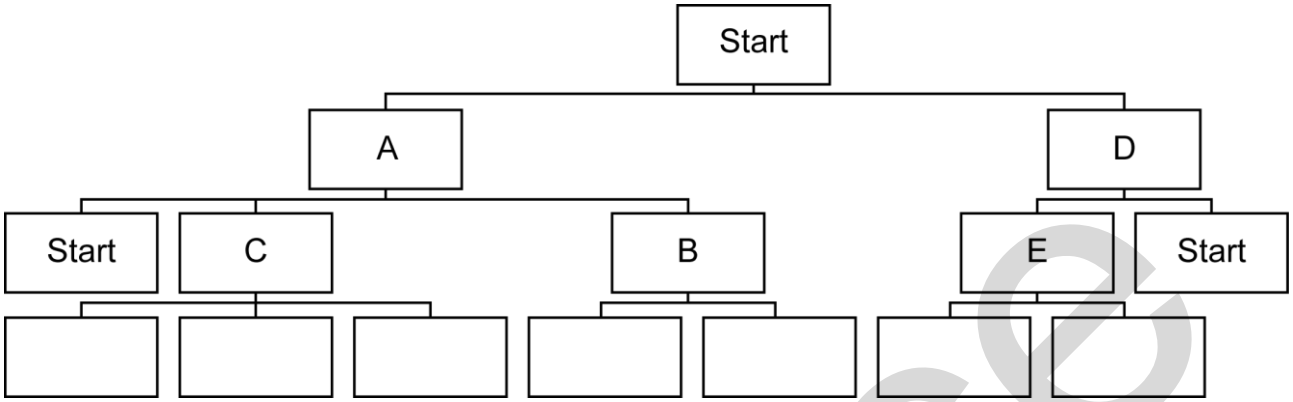

Fig 6.2

(i) The 'Start' nodes on level three are not expanded again as this is a repeat, 'Start' has already been expanded.

Write the place names in the boxes in Fig 6.2, to complete the fourth level of the tree structure for the map shown in Fig 6.1.

[3]

(ii) Explain why the tree in Fig 6.2 is **not** a binary tree.

.....

..... **[1]**

(c) The programmer has decided to use a graph instead of a tree structure.

(i) Describe what is meant by a graph structure.

.....

.....

.....

..... **[2]**

(ii) The pseudocode below shows part of an algorithm which uses a queue to traverse the graph breadth-first. Complete the missing elements of the algorithm.

```
markAllVertices (notVisited)
createQueue()
start = ..............
markAsVisited(..............)
pushIntoQueue(start)
while QueueIsEmpty() == ............
    currentNode = removeFromQueue()
    while allNodesVisited() == false
        markAsVisited(..............)
        //following sub-routine pushes all nodes connected to
        //currentNode AND that are unvisited
        pushUnvisitedAdjacents()
    endwhile
endwhile
```

[4]

(d) Fig 6.3 is a graph representation of the places that the travelling salesman visits. Using this graph, show how Dijkstra's algorithm would find the shortest path from place A to place F.

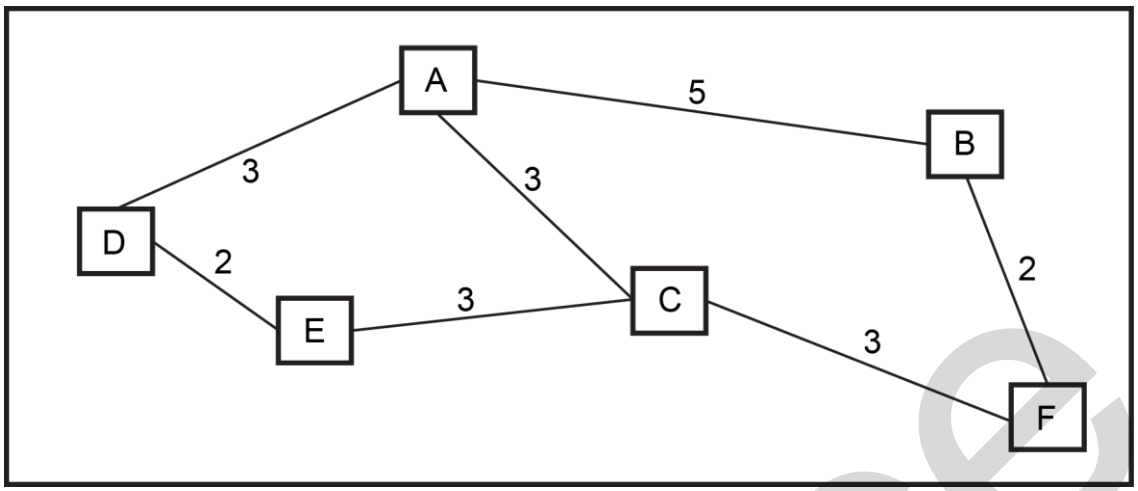

Fig 6.3

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... **[6]**

Section B

Answer **all** questions

7 Four in a Row is a game where two players drop coloured discs into a grid, with the aim to get four of their own colour in a row.
Each player is given a set of coloured discs, red (R) or yellow (Y). The players take it in turns to drop their disc into a column in the grid. The disc drops down to the lowest available space in that column.

The grids below (Fig 7.1 and 7.2) show what happens when the yellow player drops a disc into column 2:

Before

	0	1	2	3	4	5	6
0							
1							
2							
3		R	Y	Y			
4		Y	R	R	Y		
5	R	Y	R	R	Y	R	

Fig 7.1

After

	0	1	2	3	4	5	6
0							
1							
2			Y				
3		R	Y	Y			
4		Y	R	R	Y		
5	R	Y	R	R	Y	R	

Fig 7.2

The game continues until one player has got four discs of their colour in a straight row in any direction i.e. vertical, horizontal, or a diagonal.

(a) A programmer is going to use decomposition to help produce the game.

(i) Explain how decomposition can be used in the design of the game Four in a Row.

.....

.....

.....

..... **[2]**

(ii) The program will allow the players to take it in turns to make a move. Each move will be checked to ensure it is valid (i.e. the column is not already full). After each move the program will check if that player has won by checking the horizontal, vertical and diagonal positions to confirm if that player has four discs in a row.

The programmer has developed a top-down design for the program as shown in the structure diagram Fig 7.3.

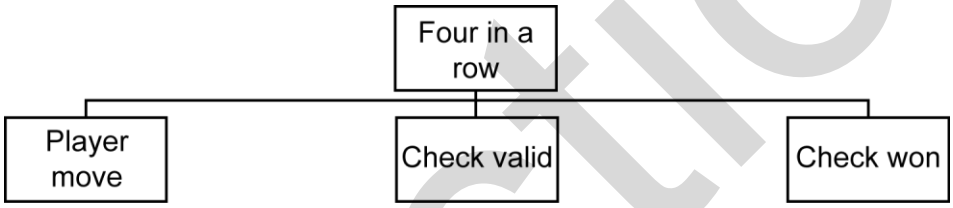

Fig 7.3

Add **one** further level to the structure diagram, by dividing the sub-modules 'Player move', 'Check valid' and 'Check won' into further sub-modules.

[3]

(a) (iii) The structured design for this program makes use of pipelining. Describe **one** example of where pipelining could be used in this program.

.....

.....

.....

..... **[2]**

(b) A 2-dimensional array, `grid`, is used to hold the game grid.

Using pseudocode, write a function that takes as input the player whose turn it is, and the column number they select as their turn. The function either:

- returns 999 (i.e. the column is already full), or
- stores the player's move in the array and returns the row the disc has been placed in.

Annotate your pseudocode with comments to show how it solves the problem.

[6]

(c) After a player makes their move, the program needs to check if that player has won (i.e. the player has four discs in a row).

Subroutines have already been written to check if the player has won vertically, or diagonally.

Using pseudocode, write a procedure that reads appropriate parameters and checks if the player has won horizontally. If the player has won, display an appropriate message identifying which player has won.

[6]

(d) (i)* The programmer is writing a new version of the game, where each player removes one disc from the bottom row of the grid before a new move is made.

In the example below, player R removes one disc from column 2 (Before) and places one in column 4 (After).

Before

	0	1	2	3	4	5	6
0							
1							
2							
3		R	Y	Y			
4		Y	R	R	Y		
5	R	Y	R	R	Y	R	

Fig 7.4

After

	0	1	2	3	4	5	6
0							
1							
2							
3		R		Y	R		
4		Y	Y	R	Y		
5	R	Y	R	R	Y	R	

Fig 7.5

The programmer has to decide whether to continue to use a 2D array, or produce an array of queues.

Evaluate the use of a 2D array versus an array of queues to perform this action.

.....

.....

.....

.....

.....

.....

.....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... **[9]**

(d) (ii) Explain why a stack would **not** be an appropriate data structure for this revised game.

....

....

....

.... **[2]**

(d) (iii) A procedure needs to be written to remove the disc from the chosen column. The procedure will:

- have the column the disc is being removed from as a parameter
- move each disc in that column down to the bottom of the grid
- replace the top space with an empty string ("")

Complete the algorithm below.

```
procedure playDisc (removeColumn)
```

[3]

(e) The programmer is adding a feature that allows a player to play against the computer.

Explain how the programmer could make use of a tree structure, using a heuristic approach in the development of the computer player.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... **[7]**

END OF QUESTION PAPER

ADDITIONAL ANSWER SPACE

If additional answer space is required, you should use the following lined page(s). The question number(s) must be clearly shown in the margin(s).