Candidate Marks Report

Series : 6 2018

This candidate's script has been assessed using On-Screen Marking. The marks are therefore not shown on the script itself, but are summarised in the table below.

Centre No :		Assessment Code :	H446
Candidate No :		Component Code :	02
Candidate Name :			

Total Marks :

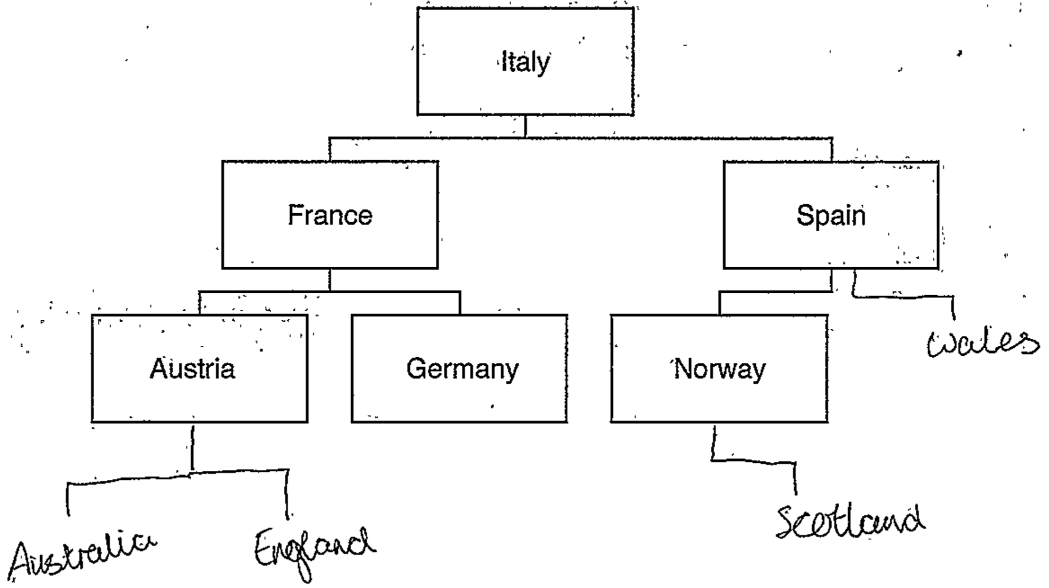
In the table below ‘Total Mark’ records the mark scored by this candidate.
‘Max Mark’ records the Maximum Mark available for the question.

Section A

Answer **all** the questions.

1 A program stores entered data in a binary search tree.

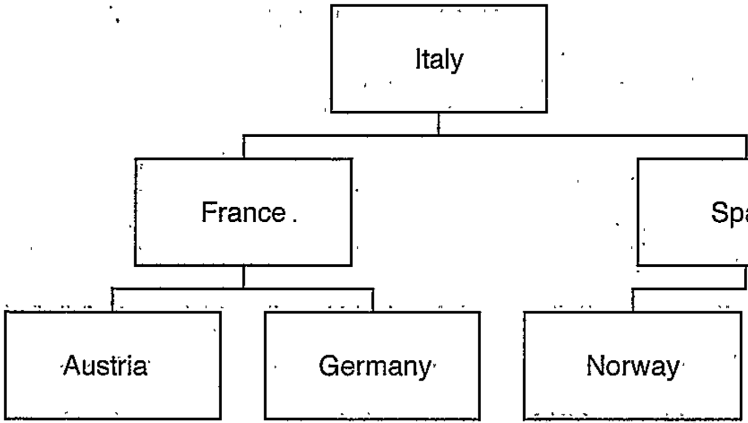
The current contents of the tree are shown:



(a) Complete the diagram to show the contents of the tree after the following data is added:

England, Scotland, Wales, Australia **[3]**

(b) Show the order of the nodes visited in a breadth first traversal on the following tree.



Italy, france, Spain, Austria, Germany, Norway

[3]

(c) A pseudocode algorithm is written to search the tree to determine if the data item "Sweden" is in the tree.

The function `currentNode.left()` returns the node positioned to the left of `currentNode`.

The function `currentNode.right()` returns the node positioned to the right of `currentNode`.

```
function searchForData(currentNode:byVal, searchValue:byVal)
    thisNode = getData(currentNode)
    if thisNode == searchValue then
        return TRUE
    elseif thisNode < searchValue then
        if currentNode.left() != null then
            return (searchForData(currentNode.left(), searchValue))
        else
            return (searchforData(currentNode.right(), searchValue))
        endif
    else
        if CurrentNode.right() ~~!= null~~ != null then
            return (searchForData(currentNode.right(), searchValue))
        else
            return false
        endif
    endif
endfunction
```

(i) Complete the algorithm.

[5]

(ii) The algorithm needs to be used in different scenarios, with a range of different trees.

Identify **two** preconditions needed of a tree for this algorithm to work.

1 The current node value

2 The search value.

[2]

BLANK PAGE

PLEASE DO NOT WRITE ON THIS PAGE

2 A company merger is joining five e-commerce retailers under one company, OCRRetail. Each retailer has a different sales system and OCRRetail wants to develop one computer system that can be used by all the retailers.

Mary's software development company has been employed to analyse and design a solution for the company.

(a) **(i)** Two computational methods (techniques used to solve a problem using computational thinking) that Mary will use are problem recognition and decomposition.

State what is meant by problem recognition and decomposition.

Recognition Identifying the problem that needs to be solved, all inputs to the solution and the proposed outputs

Decomposition Taking the problem that needs to be solved and dividing it it to sub-tasks making it clearer to solve **[2]**

(ii) State **one** additional computational method.

Abstraction **[1]**

(b) Mary plans to use data mining to generate information about OCRRetail's customers. Mary will use this information to benefit the company.

(i) Define the term 'data mining'.

It is the process of analysing large ~~amouts~~ amounts of data in order to find patterns or similarities to predict future trends. **[1]**

(ii) Identify **two** pieces of information that data mining could provide OCRRetail about sales, and state how OCRRetail could make use of this information.

1 The average demographic of their customers. This would allow them to know who to market their store to

2 What items are selling the most units in relation to other factors such as temperature or ~~day or~~ season. For example, coats might sell better in the ~~cold~~ winter. Tells them what stock to have when to maximise profitability **[4]**

(c) Mary has developed the program and is considering using performance modelling before installing the system.

(i) Define the term 'performance modelling'.

Is the ~~pros~~ process of taking a real world problem and abstracting it to a model which solutions can be tested on. **[1]**

(ii) Identify **one** way performance modelling could be used to test the new system.

They could run a simulation of an average day to identify it in what ways it is working and which it isn't. **[1]**

(d) Mary created the program as a series of sub-programs that can be reused.

Describe **one** benefit of Mary creating reusable program components.

Reusable program components could be imported into a ~~lit~~ library and called whenever they are needed, making programing quicker and testing and maitenence easier. **[2]**

3 A puzzle has multiple ways of reaching the end solution. Fig. 3 shows a graph that represents all possible routes to the solution. The starting point of the game is represented by A, the solution is represented by J. The other points in the graph are possible intermediary stages.

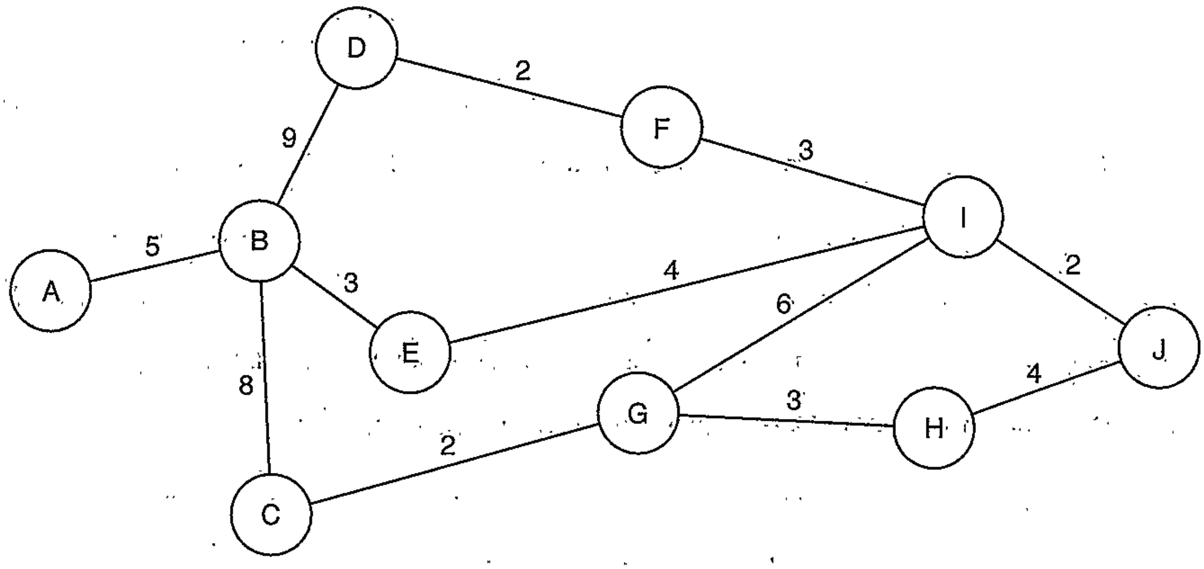


Fig. 3

(a) The graph in Fig. 3 is a visualisation of the problem.

(i) Identify **one** difference between a graph and a tree.

A tree has a root and branches coming off of it that cannot connect - not found on a graph. **[1]**

(ii) Explain how the graph is an abstraction of the problem.

The routes are represented by edges not how they are done. Each of the intermediary steps, the starting point and the solution are ~~p~~ represented by nodes instead of showing the puzzle. ~~The rou~~ A numerical weighting given to each route. **[2]**

(iii) Identify **two** advantages of using a visualisation such as the one shown in Fig. 3.

1 It allows the routes to clearly be seen without data which is irrelevant to the problem

2 An optimisation algorithm can easily be applied to identify the most efficient way to solve the problem **[2]**

(b) Demonstrate how Dijkstra's algorithm would find the shortest path to the solution in Fig. 3.

current node A

B added to queue - visited

Distance to node B : 5

A fully explored

Current node B

C, D, E added to queue - visited

~~B~~ C = 13, D = 14, E = 8 distance from A

B fully explored

Current node E

I added to queue - I visited

I distance from A = 12

E fully explored **[7]**

Current node I

G, J added to Queue -visited

G=18, J = 14 distance from A

~~J=~~

J found = TRUE

Route = A, B, E, I, J

distance 14

A → [B ; 5]

B → [D ; 9] [C ; 8] [E ; 3] [A ; 5]

C → [G ; 2] [B ; 8]

D → [F ; 2] [B ; 9]

E → [B ; 3] [I ; 4]

F → [D ; 2] [I ; 3]

G → [C ; 2] [H ; 3] [I ; 6]

H → [G ; 3] [J ; 4]

I → [E ; 4] [F ; 3] [G ; 6] [J ; 2]

J → [I ; 2] [H ; 4]

(c)* The creator of the puzzle has been told that the A* algorithm is more efficient at finding the shortest path because it uses heuristics.

Compare the performance of Dijkstra's algorithm and the A* search algorithm, making reference to heuristics, to find the shortest path to the problem.

Dijkstra's algorithm garentees that the shortest path is found but it will take a long time if there is ~~a~~ large data.

Heuristic algorithms use educated guesses and approximations to find a solution ~~the~~ to a problem which may not be perfect.

~~Either~~ Even ~~though~~ though the solution is not perfect it does work and in this situation will find a short path. The benifit of heuristic algorithms is that they take a ~~shorr~~ shorter amount of time to execute as they do not have to search every possible route. A* is a heuristic algorithm

Therefore, whenever possible Dijkstra's algorithm should be used. This will be up until the number of nodes and edges is so large that it cannot ~~cope any more~~ solve the algorithm ~~is~~ in a reasonable amount of time. Once this is the case, A* search should be used

It is up to the programmer on small tasks ~~where it~~ whether to prioritise perfection or efficiency.

[9]

(d) A computer program version of the puzzle is to be developed. A programmer will use an IDE to debug the program during development.

Describe **three** features of an IDE that help debug the program.

1 Breakpoints can be set up in the code so that the execution can be stopped at a certain point and the contents of variables and such can be analysed

2 step-through. The programmer can step-through the code watching how variables change, looking for logical errors.

3 ~~Colour coordinated code and indentation~~ Numbered lines helps the programmer to easily document the program and clearly see where they are working and can easily communicate this to others if needed.

[6]

4 A recursive function, `generate`, is shown.

```
function generate(num1:byval)
    if num1 > 10 then
        return 10
    else
        return num1 + (generate(num1 + 1) DIV 2)
    endif
endfunction
```

(a) Trace the algorithm to show the value returned when `generate(7)` is called. Show each step of your working.

Return 14

num1 = 7

num1 > 10 = FALSE

~~Output~~ Return (7 + ~~(s~~ generate(~~8~~ 7+1) DIV 2))

return (7 + (generate (8) DIV 2)) => 7 + 14 DIV 2 => 14

generate 8

num1 = 8

num1 > 10 = FALSE

Return (8 + (generate (8+1) DIV 2))

Return (8 + (generate(9) DIV 2)) => 8 + 12 DIV 2 => 14

generate 9

num1 > 10 = FALSE

Return (9 + (generate (9+1) DIV 2))

Return (9 + (generate (10) DIV 2)) => 9 + 15 DIV 2 => 12

generate 10 return (10 + (generate (11) DIV 2)) => 15 **[6]**

generate 11 return 10

(b) The parameter, `num1`, is passed by value.

Explain why the parameter was passed by value instead of by reference.

By value ~~sta~~ declares that num1 should be treated as a numerical data type, in this case it is ~~is~~ an integer. This allows it to be treated as such and it can be used in arithmetic calculations. **[2]**

(c)* Parameters can be used to reduce the use of global variables.

Compare the use of parameters to global variables in recursive functions.

Recursive functions can be called within the time when they are ~~p~~ running. This means that they can often be ~~coll~~ called may times in short succession. This is often determined by a selection that can produce new parameters. By specifying parameters at the beginning of a recursive function ~~it allows that~~ and clearly documenting them, it allows that ~~module cor of code to~~ program component to be reusable, imported into a library and called whenever needed. This makes the program much ~~eas~~ more easier to debug and maintain

~~Glo~~ It is generally considered bad programming practice to use global variables. Global variables have a ~~must~~ ~~f~~ much larger scope than parameters. Parameters are local ~~to~~ only to the module they are from and the ~~f~~ recursive function that they are therefore put in. Global variables can be accessed throughout the whole program so they ~~are difficult to~~ can be changed anywhere, making them difficult to implement into recursive functions as knowing what they contain at any ~~At~~ given point can be a problem. Also, a recursive function will constantly overwrite the global variable

Overall, it is better to use parameters for recursive functions as it will create clear, reusable program components. **[9]**

Turn over

(d) A student called Jason writes a recursive algorithm. The recursive algorithm uses more memory than if Jason had written it as an iterative algorithm.

Explain why the recursive algorithm uses more memory than the iterative algorithm.

Each time a recursive ~~algo~~ function is called in a recursive algorithm the new parametes need to be saved in the memory, this can happen many times causing a lot of ~~mo~~ memory to be used. **[2]**

5 A computer program stores data input on a stack named `dataItems`. The stack has two sub-programs to add and remove data items from the stack. The stack is implemented as a 1D array, `dataArray`.

Sub-program	Description
push()	The parameter is added to the top of the stack
pop()	The element at the top of the stack is removed

The current contents of `dataItems` are shown:

6
15
100
23

(a) Show the contents of the stack `dataItems` after each line of the following lines of code are run

```
01  push(13)
02  pop()
03  push(10)
04  push(20)
```

Line 01	Line 02	Line 03	Line 04
			20
13		10	10
6	6	6	6
15	15	15	15
100	100	100	100
23	23	23	23

[4]

(b) The main program asks a user to push or pop an item from the stack. If the user chooses 'push', the data item is added to the stack. If the user chooses "pop", the next item is removed from the stack, multiplied by 3 and output.

The main program is shown:

```
01 userAnswer = input("Would you like to push or pop an item?")
02 if userAnswer == "push" then
03     push(input("Enter data item"))
04 else
05     print(pop() * 3)
06 endif
```

(i) Before the sub-programs, `push()` and `pop()`, can add or remove items from the stack, a selection statement is used to decide if each action is possible.

Describe the decision that needs to be made in each sub-program and how this impacts the next process.

`push()` Decides if the stack is full. If it is not full, it should continue. ~~It it~~ If it is full an error should be shown

`pop()` Decides if the stack is empty. If it is not empty it should continue. ~~If~~ If it is empty an error should be shown.

[4]

(ii) The algorithm does not work when the user enters `"PUSH"` or `"Push"`. The algorithm needs to be changed in order to accept these inputs.

Identify the line number to be changed and state the change that should be made.

Line number 02

Change if userAnswer == "push" OR ~~"PUSH" OR~~ userAnswer == "PUSH" OR userAnswer == "Push" THEN

[2]

(c) The stack is implemented as a 1D array, `dataArray`.

Describe how a 1D array can be set up and used to push and pop items as a stack.

A 1D array would contain all the items in the stack. push would add the item to the end of the ~~I~~ 1D array and pop would return the last ~~it~~ item in the array.

[3]

(d) As an array, the data in `dataArray` is sorted and then searched for a specific value.

(i) The data in `dataArray` is sorted into ascending order using an insertion sort.

The current contents of `dataArray` are shown:

100	22	5	36	999	12

Show the steps of an insertion sort on the current contents of the array `dataArray`.

100 22 5 36 999 12

22 100 5 36 999 12

5 22 100 36 999 12

5 22 36 100 999 12

999 does not move

5 22 36 100 999 12

[5]

5 12 22 36 100 999

(ii) The array `dataArray` can now be searched using a binary search.

Describe the stages of a binary search on an array of size `n`.

Begin with a search interval of size n and move a pointer to the mid-point - $\frac{1}{2}n$.

If the mid-point ~~is gre~~ value is greater than the search value the mid-point and the values to the left are removed from search. Opposite if it is less.

This process of removing half of the ~~sea~~ search interval by half continues iteratively until the mid-point matches the seach value ~~of the~~ or the search iterval of 1 item does not contain the search value. Algorithmic complexity of ~~lo~~ O(logn).

[7]

If the mid-point matches with a ~~value~~ seach value then the mid-point's position is the position of the item the user was searching for. This ~~meose~~ means that, if needed, the ~~mid-point~~ value of the mid-point could be returned as the location of item

This is a divide and conquor method of searching as large sets of data (in this case half) are discarded each time

(iii) The array has 50 items.

The function, `searchItem()`, performs a linear search for a data item.

```
function searchItem(dataItem)
    for count = 0 to 49
        if dataArray[count] == dataItem then
            return(count)
        endif
    next count
    return(-1)
endfunction
```

Rewrite the function using a while loop.

```
function searchItem(dataItem)
    i = 0
    while i < 50:
        if data Array[count] = = dataItem then
            return (count)
        endif
        i = i + 1
    return(-1)
endfunction
```

[4]

Section B

Answer **all** questions.

6 Kamran is writing a program to manipulate the data for a set of items.

For each item, the program needs to store:

- Item name (e.g. Box)
- Cost (e.g. 22.58)
- Date of arrival (e.g. 1/5/2018)
- Transferred (e.g. true)

The items are added to a queue for processing.

The queue is defined as a class, `itemQueue`.

```
itemQueue
--------------------------
theItems[10] : Items
head : Integer
tail : Integer
numItems : Integer
--------------------------
constructor
enqueuer()
dequeuer()
setnumItems()
getnumItems()
```

The `head` attribute points to the first element in the queue. The `tail` attribute points to the next available space in the queue. The `numItems` attribute states how many items are currently in the queue.

(a) The data about the items can be stored using either a record structure, or as objects of a class.

(i) Explain the similarities and differences between a record and a class.

Record and class are both complex data types ~~but~~ that can contain primitive data types. Both are not restricted to contain just one data type. A record cannot be changed dynamically throughout the program where a class ~~a~~ can. Class can include operations that can be performed on the data **[3]**

(ii) Kamran chooses to use a record structure to store the data about the items.

Record structures may be declared using the following syntax:

```
recordStructure recordstructurename
    fieldname : datatype
    ...
endRecordStructure
```

Complete the pseudocode to declare a record called `items`.

recordStructure Items

itemName : String

Cost : Currency

date ~~Date of arrival~~ dateofarrival : Date

transferred : boolean

endRecordStructure

[5]

(iii) New records may be created using the following syntax:

```
recordidentifier : recordstructurename
recordidentifier.fieldname = data
...
```

Write a programming statement to create a new item, using the identifier 'box1', with the item name "Box", the cost 22.58, date of arrival 1/5/2018 and transferred true.

recordidentifier : Items

recordidentifier.fieldname = 'box1'

recordidentifier.itemname = 'box'

recordidentifier.Cost = 22.58

recordidentifier.dateofarrival = 1/5/~~2~~2018

recordidentifier.transferred = TRUE.

[3]

(b) The array, `theItems`, stores the items in the queue. When the tail of the queue exceeds the last element in the array, it adds a new item to the first element if it is vacant.

For example, in the following queue, the next item to be added would be placed at index 0.

Index	0	1	2	3	4	5	6	7	8	9
Element				Data	Data	Data	Data	Data	Data	Data

(i) Define the term 'queue'.

A queue is an array ~~of data~~ in which ~~is removed~~ data is processesed with a first in first out policy. Items are added to the ~~front~~ back and removed from front **[2]**

(ii) The attributes in `itemQueue` are all declared as private.

Explain how a private attribute improves the integrity of the data.

Since ~~they a~~ they are private they cannot ~~e~~ be edited dynamically. This ensures that no changes can be made that will contradict any other data on the system **[2]**

(iii) The constructor method creates a new instance of `itemQueue` and sets the `head`, `tail` and `numItems` attributes to 0.

Write an algorithm, using pseudocode or program code, for the constructor including the initialisation for all attributes.

```
 itemQueue                          head = 0
theItems [10] : ~~Itr~~ Items       tail = 0
head ~~=~~ : ~~0~~ Integer          numItems = 0
tail ~~=~~ : ~~0~~ Integer
numItems : ~~0~~ Integer
```

[2]

~~head~~

(iv) The `enqueue` method:

- takes as a parameter the item to insert in the queue
- checks if the queue is full
- reports an error and returns `false` if the queue is full
- does the following if the queue is not full:
    - adds the item to the array at the tail position and adjusts the pointer(s)
    - returns `true`

The attribute `numItems` stores the number of items currently in the queue.

Write an algorithm, using pseudocode or program code, for the `enqueue` method.

```
function enqueue (~~item~~ new)
  If len~~g~~(the Items) = 10 then
    Output ('~~Queue full~~' Error: Queue full, item has not
            been added')
  ~~else:~~ return false
  else
    the Items[tail] = new
    tail = tail + 1
    return true
  endif
```

[6]

(v) Write a programming statement to declare an instance of `itemQueue` called `myItems`.

~~x =~~ myItems = itemQueue ()

[1]

(vi) Write a procedure, `insertItems()`, to ask the user to input the data for an item. The item is then added to the queue `myItems`. The user is continually asked to input data items until the queue is full.

~~function~~

~~Fns~~ = ~~insertItems()~~ ~~tail = 0~~

procedure insertItems()

    i = 0

    while i < 10 do

        ~~data = t~~

        INPUT data ('Please input data')

        myItems [~~tail~~] = data

        i = i + 1

    OUTPUT ('Queue full')

[5]

(vii) When the main program ends, the items and the queue no longer exist.

Describe how Kamran could amend the program to make sure the items and queue still exist and are used the next time the program is run.

Store them to a permenant memory location outside of the program. Such as exporting it to a file which can be saved onto the hardisk or any other secondary storage. These can then be retrieved the next time it is run. **[2]**

(c)* Kamran wants to expand the program to allow it to handle up to 100,000,000 items and to allow him to search for data about items. Kamran is worried that the increase in the number of items will cause a decrease in the performance of the program. He decides to investigate the benefits of caching and concurrent processing.

Evaluate the use of caching and concurrent processing in this scenario and make a recommendation to Kamran.

Caching would allow commonly searched for items to be stored in a different memory location so that they can be found more efficiently. ~~for~~

Concurrent processing can be done in ~~no~~ many different ~~wasy~~ ways. Two of the most popular are ~~pipe~~ pipelinging and ~~incre~~ increasing the number of cores. Though changes may have to be made

Here is my reccommendation. ~~Us~~ Analyse the system for the most commonly searched items and cache them. If the cost is worth it (probabily will be) increase the ~~A~~ number of cores and adapt software to take adventage of this. Implement pipelining.

Pipelining is ~~the~~ when the processor is able to overlap the stages of the fetch-decode-execute cycle.

[9]

END OF QUESTION PAPER

PLEASE DO NOT WRITE ON THIS PAGE

OCR
Oxford Cambridge and RSA