Cambridge Assessment International Education

# Cambridge International AS & A Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER



**COMPUTER SCIENCE** **9608/43**

Paper 4 Further Problem-solving and Programming Skills **October/November 2021**

**2 hours**

You must answer on the question paper.

No additional materials are needed.

**INSTRUCTIONS**

- Answer **all** questions.
- Use a black or dark blue pen.
- Write your name, centre number and candidate number in the boxes at the top of the page.
- Write your answer to each question in the space provided.
- Do **not** use an erasable pen or correction fluid.
- Do **not** write on any bar codes.
- You may use an HB pencil for any diagrams, graphs or rough working.
- Calculators must **not** be used in this paper.

**INFORMATION**

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [ ].
- No marks will be awarded for using brand names of software packages or hardware.

This document has **20** pages. Any blank pages are indicated.


© UCLES 2021

**1** Sandy is writing a program to process data in a stack. The stack is implemented as a 1D array, `DataStack`, which has up to 100 elements.

The function `Push(Value)` stores `Value` on the stack and returns `TRUE` if `Value` was added to the stack, or `FALSE` if the stack is full.

The function `Pop()` returns the item at the top of the stack, or returns `-1` if the stack is empty.

`DataStack` and `TopPointer` are declared as global.

**(a)** Show the state of `DataStack` and its pointer after the following functions are executed on the current contents.

```
Pop()

Pop()

Push(19)

Pop()

Push(50)
```

**TopPointer** 3

| Index | Data |
|---|---|
| [7] | |
| [6] | |
| [5] | |
| [4] | |
| [3] | 8 |
| [2] | 6 |
| [1] | 20 |
| [0] | 10 |

[2]

**(b)** Write **program code** for the function `Pop()`.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [5]

**(c)** Sandy has also used a queue in her program.

Describe the ways in which a queue differs from a stack.

.....

.....

.....

.....

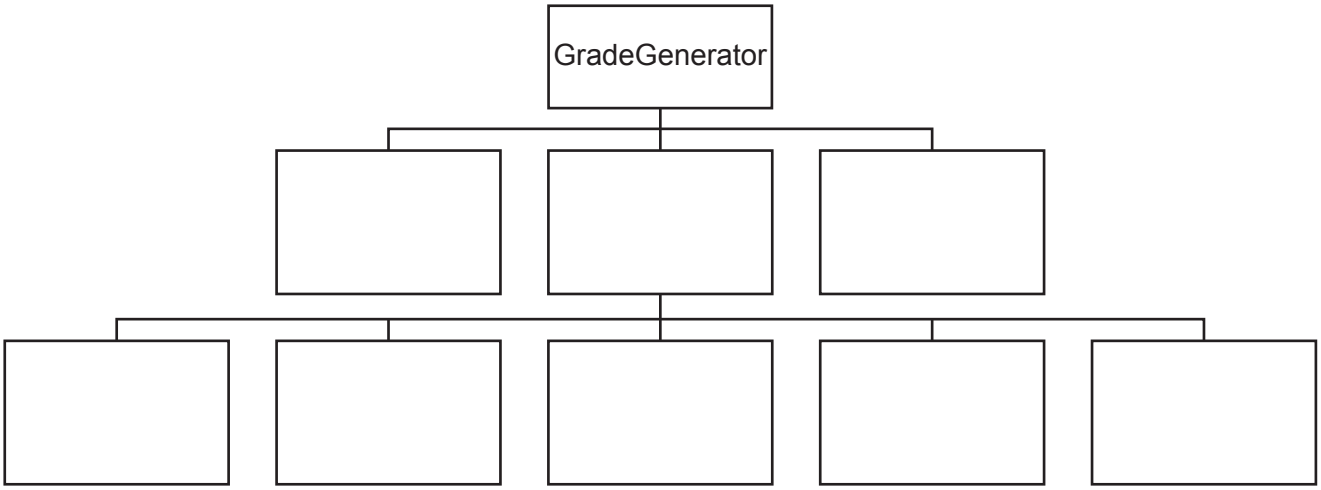
.....

..... [2]

**2** A grade generator program takes the mark a student obtained in a test as input.

The program calculates and outputs the grade that matches the mark. The grade is either A, B, C, D or U.

Complete the following JSP structure diagram for the grade generator program.




[4]

**3** The following pseudocode algorithm performs a binary search on the sorted array `ThisArray`.

The algorithm returns either the location of `SearchItem` in the array, or `-1` if `SearchItem` is not in the array.

The function `DIV` returns the integer value of the division, for example, `11 DIV 2` returns `5`.

Complete the algorithm by writing the missing pseudocode statements.

```
FUNCTION BinarySearch(ThisArray[], LowerBound, UpperBound,
                      SearchItem : INTEGER) RETURNS INTEGER

DECLARE Flag : BOOLEAN

DECLARE Mid : INTEGER

Flag ← -2

WHILE Flag <> -1

   Mid ← LowerBound + ((UpperBound - LowerBound) DIV 2)

   IF ..... < .....

      THEN

         RETURN .....

      ELSE

         IF ThisArray[Mid] > SearchItem

            THEN

              UpperBound ← Mid .....

            ELSE

               IF ThisArray[Mid] < SearchItem

                  THEN

                     LowerBound ← Mid .....

                  ELSE

                     RETURN .....

               ENDIF

         ENDIF

   ENDIF

ENDWHILE

ENDFUNCTION
```

[6]

**4** Teachers in a school may work on Mondays, Tuesdays and Wednesdays. There are three time slots on each day: time slot 1, time slot 2 and time slot 3.

A teacher is either busy or free.

The school is using a declarative language to write a program to record which teachers are busy in each time slot on each day.

The following knowledge base is used:

```
01 teacher(james).
02 teacher(jill).
03 teacher(karl).
04 teacher(kira).
05 day(monday).
06 day(tuesday).
07 day(wednesday).
08 timeSlot(1).
09 timeSlot(2).
10 timeSlot(3).
11 busy(james, monday, 1).
12 busy(james, tuesday, 2).
13 busy(karl, monday, 1).
14 busy(kira, wednesday, 3).
```

These clauses have the following meaning:

| Clause | Explanation |
|---|---|
| 01 | James is a teacher |
| 05 | Monday is a day |
| 08 | 1 is a time slot |
| 11 | James is busy in time slot 1 on Monday |

**(a)** More facts need to be included.

Fred is a teacher who is busy in time slot 1 on Tuesday.

Write additional clauses for these facts.

15 .....

16 .....

[2]

**(b)** Additional clauses are needed to identify whether Jill is busy in time slot 1 on Monday, Tuesday, or Wednesday.

Write these additional clauses.

```
17 .....

18 .....

19 .....
```

[2]

**(c)** Write a goal, using the variable `X`, to find all the teachers who are busy in time slot 3 on Monday.

.....

..... [1]

**(d)** Write a rule to find whether a teacher `X` is free in a specific time slot `Y` on day `Z`.

```
IsTeacherFree(X, Z, Y)

IF
```

.....

.....

.....

..... [4]

**5** The recursive algorithm for the `Recursion()` function is defined in pseudocode as follows:

```
FUNCTION Recursion(A, B : INTEGER) RETURNS INTEGER
   IF A <= 100
      THEN
         RETURN 1
      ELSE
         IF A > B
            THEN
               RETURN 5 + Recursion(A - 1, B)
            ELSE
               RETURN 10 + Recursion(A – 10, B)
         ENDIF
   ENDIF
ENDFUNCTION
```

**(a)** The function is called with the following pseudocode statement:

```
OUTPUT Recursion(104, 102)
```

Dry run the function and complete the trace table. Give the output the program will produce.

**Trace table:**

| Function call | A | B | Return value |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

**Output =** .....

Working .....

.....

.....

.....

.....

.....
[4]

**(b)** Rewrite the function `Recursion()` in **pseudocode**, using an **iterative algorithm**.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

**6** Kobi is writing an application that uses a record structure to store data.

**(a) (i)** Describe what is meant by a **record structure**.

.....

.....

.....

..... [2]

**(ii)** The record structure stores the unique ID number (a whole number), first name and last name of a customer.

Write a **pseudocode** declaration for the record structure `CustomerData`.

.....

.....

.....

.....

.....

.....

..... [2]

**(b)** Kobi's application stores the records in a random access file.

The function `StoreRecord()`:

- takes a customer record as a parameter
- uses the function `CustomerHash()` to calculate and return the hash value for its parameter
- stores the customer record in the returned hash value address.

Assume there are no collisions.

Complete the following pseudocode algorithm to write a new record to the random access file.

```
PROCEDURE  StoreRecord(NewData : ……)
    HashValue ← CustomerHash(NewData.CustomerID)
    Filename ← "CustomerRecords.dat"
    OPENFILE Filename FOR ……
    SEEK Filename, ……
    PUTRECORD Filename, ……
    …… Filename
ENDPROCEDURE
```

[5]

**(c)** Identify **two** typical features of a debugger **and** describe how Kobi could use each one during the development of the application.

Feature 1 .....

.....

.....

.....

Feature 2 .....

.....

.....

.....

[4]

**(d)** Give **one** benefit and **one** drawback of Kobi using a program generator whilst developing his application.

Benefit .....

.....

Drawback .....

.....

[2]

**7** Sonya is writing a computer program that requires a user input. The user should input an integer between 1 and 100. Sonya wants to use exception handling.

**(a)** Explain the reasons why Sonya should use exception handling in her program.

.....

.....

.....

..... [2]

**(b)** Write **program code** to read in the number from the user and raise an exception if the data is not valid.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

..... [3]

**(c)** Give **two other** examples of where exception handling can be used in a program.

1 .....

2 .....

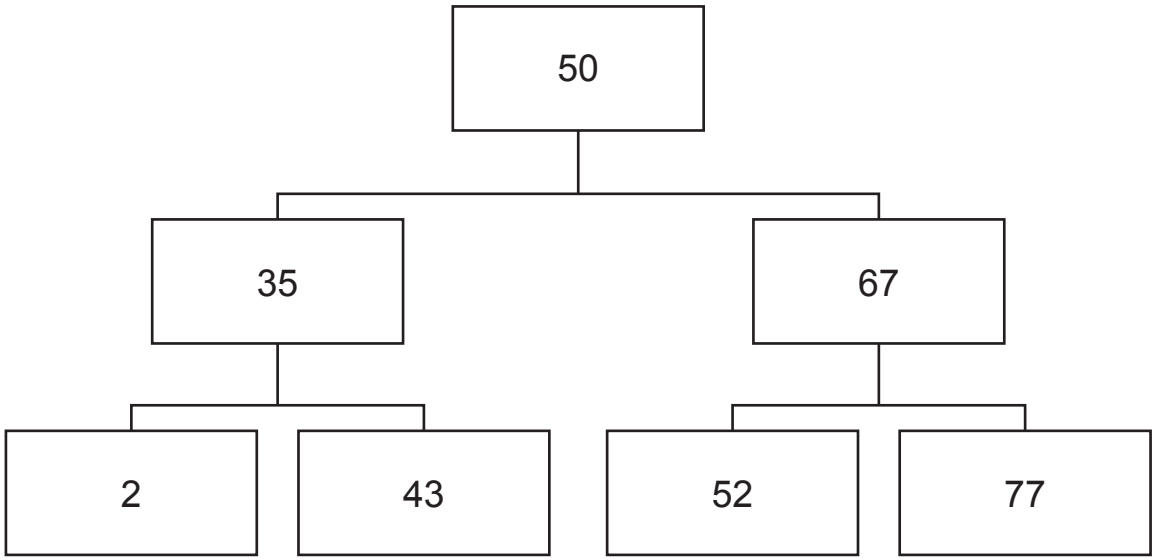
[2]

**8** Data entered into a computer is stored in an ordered binary tree.

The binary tree is stored in a 2D array, `BinaryTree`.

The first element of the array is index 0.

**(a)** The current contents of the binary tree are:




Complete the `LeftPointer` and `RightPointer` values in the following table for the binary tree shown.

A null pointer is represented by `-1`.

**RootNode** 0

| Index | LeftPointer | Data | RightPointer |
|---|---|---|---|
| [0] | | 50 | |
| [1] | | 67 | |
| [2] | | 77 | |
| [3] | | 35 | |
| [4] | | 2 | |
| [5] | | 43 | |
| [6] | | 52 | |
| [7] | | | |
| [8] | | | |
| [9] | | | |
| [10] | | | |

[2]

**(b)** A post-order tree traversal outputs the left node, then the right node, then the root node.

In the tree given in **part (a)**, the post-order tree traversal would output:

```
2     43     35     52     77     67     50
```

Complete the following recursive pseudocode algorithm `PostOrder()`.

```
PROCEDURE PostOrder(…… : INTEGER)
   IF BinaryTree[RootNode, 0] <> -1
     THEN
        …… (BinaryTree[RootNode, ……])
   ENDIF
   IF BinaryTree[RootNode, 2] <> -1
     THEN
        …… (BinaryTree[RootNode, 2])
   ENDIF
   OUTPUT BinaryTree[RootNode, ……]
ENDPROCEDURE
```

[5]

**BLANK PAGE**

**9** A program uses a hashing algorithm to store data in the global array, `StoredData`.

The first element of the array is index 0. The array has 10 000 integer elements.

**(a)** Write a **pseudocode** declaration for the array `StoredData` **and** initialise each element to `-1`.

.....

.....

.....

..... [3]

**(b)** The hashing algorithm calculates the remainder after dividing the data by 1000, and then adds 6 to it.

The function `AddItem()` takes the data as a parameter. It calculates the index to store the data using the hashing algorithm.

If there is a collision, the function:

- checks the next index until it finds an index that does not have data in it
- continues to search from the start of the array, if it reaches the end of the array.

The function returns `TRUE` if the item was successfully added, and `FALSE` if the array is full.

Write **program code** for the function `AddItem()`.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

**BLANK PAGE**

© UCLES 2021

**BLANK PAGE**

© UCLES 2021

**BLANK PAGE**

Permission to reproduce items where third-party owned material protected by copyright is included has been sought and cleared where possible. Every reasonable effort has been made by the publisher (UCLES) to trace copyright holders, but if any items requiring clearance have unwittingly been included, the publisher will be pleased to make amends at the earliest possible opportunity.

To avoid the issue of disclosure of answer-related information to candidates, all copyright acknowledgements are reproduced online in the Cambridge Assessment International Education Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cambridgeinternational.org after the live examination series.

Cambridge Assessment International Education is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

© UCLES 2021