Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME: **Model Answers**

CENTRE NUMBER | | | | | |

CANDIDATE NUMBER | | | | |

9601387009

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2021**

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

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.
- The insert contains all the resources referred to in the questions.

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

Refer to the **insert** for the list of pseudocode functions and operators.

1 **(a)** A programmer applies decomposition to a problem that she has been asked to solve.

Describe decomposition.

Decomposition is breaking down a complex task into smaller sub-tasks.
It makes the task more manageable and easy to carry out.
..... [2]

(b) The following pseudocode assigns a value to an element of an array:

```
ThisArray[n] ← 42
```

Complete the following table by writing the answer for each row.

	Answer
The number of dimensions of `ThisArray`	**1**
The technical terms for minimum and maximum values that the variable `n` may take	**Lower bound, upper bound**
The technical term for the variable `n` in the pseudocode statement	**Index**

[3]

(c) Complete the pseudocode expressions so that they evaluate to the values shown.

Any functions and operators used must be defined in the **insert**.

Expression	**Evaluates to**
..... **ASC** `('C')`	`67`
`2 *` **STR_TO_NUM** `("27")`	`54`
..... **INT** ..... `(27 /` ..... **2** ..... `)`	`13`
`"Sub" &` ..... **MID** ..... `("Abstraction" ,` ..... **4** ..... **,** ..... **5** ..... `)`	`"Subtract"`

[4]

(d) Evaluate the expressions given in the following table. The variables have been assigned values as follows:

```
PumpOn ⟵  TRUE
PressureOK ⟵ TRUE
HiFlow ⟵ FALSE
```

Expression	Evaluates to
`PressureOK AND HiFlow`	**FALSE**
`PumpOn OR PressureOK`	**TRUE**
`NOT PumpOn OR (PressureOK AND NOT HiFlow)`	**TRUE**
`NOT (PumpOn OR PressureOK) AND NOT HiFlow`	**FALSE**

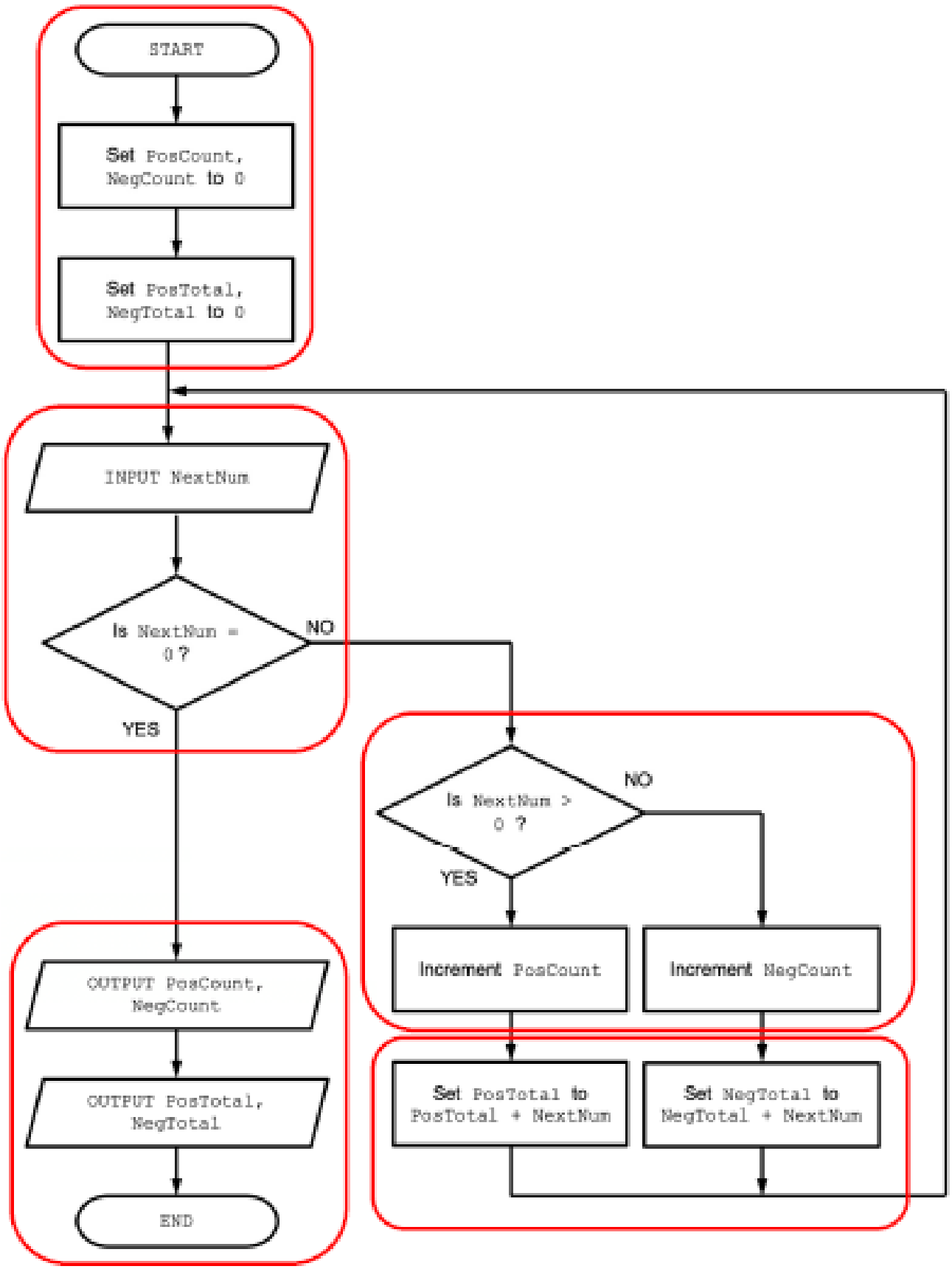
[2]

2 **(a)** An algorithm will:

1. input an integer value
2. jump to step 6 if the value is zero
3. sum and count the positive values
4. sum and count the negative values
5. repeat from step 1
6. output the two sum values and the two count values.

Draw a program flowchart on the following page to represent the algorithm.

Note that variable declarations are not required in program flowcharts.



One mark for each outlined group.

[5]

(b) A software company is working on a project to develop a website for a school.

The school principal has some ideas about the appearance of the website but is unclear about all the details of the solution. The principal would like to see an initial version of the website.

(i) Identify a life cycle method that would be appropriate in this case.

Give a reason for your choice.

Life cycle method ...**Iterative**

Reason ....**Since Iterative method produces a working model at an early stage, the principal would be happy.**

.....

.....

.....

[2]

(ii) The website project has progressed to the design stage.

State **three** activities that will take place during the design stage of the program development life cycle.

1 Decide on the data structures that would become necessary

2 Relevant algorithms would be written using flowcharts and pseudocode.

3 A suitable programming language would be chosen.

[3]

BLANK PAGE

3 A programmer is writing a program to help manage clubs in a school.

Data will be stored about each student in the school and each student may join up to three clubs.

The data will be held in a record structure of type `Student`.

The programmer has started to define the fields that will be needed as shown in the following table.

Field	Typical value	Comment
`StudentID`	`"CF1234"`	Unique to each student
`Email`	`"Carmen47@xyzmail.com"`	Contains letters, numbers and certain symbols
`Club_1`	`1`	Any value in the range 1 to 99 inclusive
`Club_2`	`14`	Any value in the range 1 to 99 inclusive
`Club_3`	`27`	Any value in the range 1 to 99 inclusive

(a) (i) Write pseudocode to declare the record structure for type `Student`.

```
TYPE Student
    DECLARE StudentID : STRING
    DECLARE Email : STRING
    DECLARE Club_1 : INTEGER
    DECLARE Club_2 : INTEGER
    DECLARE Club_3 : INTEGER
ENDTYPE
```

..... [3]

(ii) A 1D array `Membership` containing 3000 elements will be used to store the student data.

Write pseudocode to declare the `Membership` array.

```
DECLARE Membership : ARRAY [1:3000] OF Student
```

..... [2]

(iii) Some of the elements of the array will be unused.

Give an **appropriate** way of indicating an unused array element.

Unused elements can be indicated by assigning an empty string to the StudentID field.

..... [1]

(iv) Some students are members of less than three clubs.

State **one** way of indicating an unused club field.

A number like 0 which is outside the range (1-99) can be used to indicate an unused club field.

..... [1]

(b) A procedure `GetIDs()` will:

- prompt and input the number of a club
- output the `StudentID` of all the students who are members of that club
- output a count of all students in the given club.

Write pseudocode for the procedure `GetIDs()`.

```
PROCEDURE GetIDs()
    DECLARE Index : INTEGER
    DECLARE ClubID, Count : INTEGER

    OUTPUT "Please Input Club Number: "
    INPUT ClubID

    Count ← 0

    FOR Index ← 1 TO 3000
        IF Membership[Index].Club_1 = ClubID OR __
        Membership[Index].Club_2 = ClubID OR __
        Membership[Index].Club_3 = ClubID THEN
            Count ← Count + 1
            OUTPUT Membership[Index].StudentID
        ENDIF
    NEXT Index
    OUTPUT "There are ", Count, " Students in the club", ClubID
ENDPROCEDURE
```

..... [7]

4 The following is a procedure design in pseudocode.

Line numbers are given for reference only.

```
10  PROCEDURE Check(InString : STRING)
11     DECLARE Odds, Evens, Index : INTEGER
12
13     Odds ← 0
14     Evens ← 0
15     Index ← 1
16
17     WHILE Index <= LENGTH(InString)
18        IF STR_TO_NUM(MID(InString, Index, 1)) MOD 2 <> 0 THEN
19           Odds ← Odds + 1
20        ELSE
21           Evens ← Evens + 1
22        ENDIF
23        Index ← Index + 1
24     ENDWHILE
25
26     CALL Result(Odds, Evens)
27  ENDPROCEDURE
```

(a) Complete the following table by giving the answers, using the given pseudocode.

	Answer
A line number containing a variable being incremented	**19**
The type of loop structure	**Pre-condition**
The number of functions used	**3**
The number of parameters passed to `STR_TO_NUM()`	**1**
The name of a procedure other than `Check()`	**Result**

[5]

(b) The pseudocode includes several features that make it easier to read and understand.

Identify **three** of these features.

1 Using meaningful variables

2 Indentation of statements

3 Keywords given in upper-case letters

[3]

(c) (i) The loop structure used in the pseudocode is not the most appropriate.

State a more appropriate loop structure **and** justify your choice.

Loop structure - **Count-controlled loop**

Justification – **The number of repetitions (length of the string) is known.**

.....

[2]

(ii) The appropriate loop structure is now used. Two lines of pseudocode are changed and two lines are removed.

Write the line numbers of the two lines that are removed.

Lines 15 and 23

..... [1]

5 A company has several departments. Each department stores the name, email address and the status of each employee in that department in its own text file. All text files have the same format.

Employee details are stored as three separate data strings on three consecutive lines of the file. An example of the first six lines of one of the files is as follows:

File line	Comment
1	First employee name
2	First email address
3	First employee status
4	Second employee name
5	Second email address
6	Second employee status

A procedure `MakeNewFile()` will:

- take three parameters as strings:
  - an existing file name
  - a new file name
  - a search status value
- create a new text file using the new file name
- write all employee details to the new file where the employee status is **not** equal to the search status value
- count the number of sets of employee details that were in the original file
- count the number of sets of employee details that were written to the new file
- produce a summary output.

An example summary output is as follows:

```
File Marketing contained 54 employee details
52 employee sets of details were written to file NewMarketingList
```

(a) Write pseudocode for the procedure `MakeNewFile()`.

```
PROCEDURE MakeNewFile(OldFile, NewFile, Status : STRING)
    DECLARE Line1, Line2, Line3 : STRING
    DECLARE NumCopied, NumRecs : INTEGER

    NumRecs ← 0
    NumCopied ← 0

    OPENFILE OldFile FOR READ
    OPENFILE NewFile FOR WRITE

    WHILE NOT EOF(OldFile)
        READFILE OldFile, Line1
        READFILE OldFile, Line2
        READFILE OldFile, Line3
```

```
            NumRecs ← NumRecs + 1
            IF Line3 <> Status THEN
                  WRITEFILE NewFile, Line1
                  WRITEFILE NewFile, Line2
                  WRITEFILE NewFile, Line3
                  NumCopied ← NumCopied + 1
            ENDIF
      ENDWHILE
      OUTPUT "File " , OldFile , " contained " , NumRecs ,__
      " employee details"
      OUTPUT Numcopied , " employee sets of details were __
      written to file", NewFile
      CLOSEFILE OldFile
      CLOSEFILE NewFile
ENDPROCEDURE
```

..... [7]

(b) An alternative format could be used for storing the data.

A text file will still be used.

(i) Describe the alternative format.

Can store all three items on one line.

..... [1]

(ii) State **one** advantage **and one** disadvantage of the alternative format.

Advantage **Number of file read/write operations will be less.**

Disadvantage **Instructions to combine the three lines as well as to retrieve the lines separately, would be complicated.**

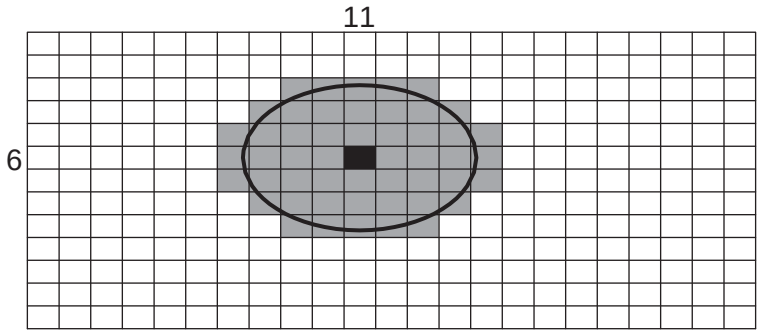
.....
[2]

6 A mobile phone has a touchscreen. The screen is represented by a grid, divided into 800 rows and 1280 columns.

The grid is represented by a 2D array `Screen` of type `INTEGER`. An array element will be set to 0 unless the user touches that part of the screen.

Many array elements will set to 1 by a single touch of a finger or a stylus.

The following diagram shows a simplified touchscreen. The dark line represents a touch on the screen. All grid elements that are wholly or partly inside the outline will be set to 1. These elements are shaded.
The element shaded in black represents the centre point of the touch.



A program is needed to find the coordinates (the row and column) of the centre point. The centre point on the diagram shown is row 6, column 11.

Assume:

- the user may only touch one area at a time
- screen rotation does not affect the touchscreen.

The programmer has decided to use global values `CentreRow` and `CentreCol` as coordinate values for the centre point.

The programmer has started to define program modules as follows:

Module	Description
`FirstRowSet()`	• Searches for the first row that has an array element set to 1 • Returns the index of that row (1 is the first row) • Returns −1 if there are no elements set to 1
`LastRowSet()`	• Searches for the last row that has an array element set to 1 • Returns the index of that row • Returns −1 if there are no elements set to 1
`FirstColSet()`	• Searches for the first column that has an array element set to 1 • Returns the index of that column (1 is the first column) • Returns −1 if there are no elements set to 1
`LastColSet()`	• Searches for the last column that has an array element set to 1 • Returns the index of that column • Returns −1 if there are no elements set to 1

(a) Write efficient pseudocode for the module `FirstRowSet()`.

```
FUNCTION FirstRowSet() RETURNS INTEGER
    DECLARE Row, Col : INTEGER
    DECLARE Found : BOOLEAN

    // array is 1280 × 800
    Row ← 1
    Found ← FALSE
    WHILE Row <= 800 AND Found = FALSE // top to bottom
        Col ← 1
        WHILE Col <= 1280 AND Found = FALSE // left to right
            IF Screen[Row,Col] = 1 THEN
                Found ← TRUE // end function as soon as first
                             // found
            ENDIF
            Col ← Col + 1
        ENDWHILE
        Row ← Row + 1
    ENDWHILE

    IF Found = FALSE THEN // nothing found
        Row ← 0
    ENDIF
    RETURN Row - 1

ENDFUNCTION
```

..... [7]

(b) Describe a feature of your solution to **part (a)** that indicates the pseudocode represents an efficient algorithm.

In my solution I use a flag to exit the loop when a screen element with value 1 is found, without going through to the end of the loop.

..... [2]

(c) The programmer decides to produce a **single** search module `FindSet()`, which will be able to perform each of the individual searches performed by the first four modules in the table.

(i) Outline the changes needed to convert one of the existing modules into this single module.

He should use a parameter to pass the type of search to the module. Search module will use global variables.

..... [2]

(ii) Give one possible advantage **and** one possible disadvantage of combining the four searches into a single module.

Advantage **Since there is only one module, it will be easy to do any changes.**

Disadvantage If it is only one module, work cannot be assigned to multiple programmers.

.....
[2]

(d) An additional module `GetCentre()` is defined as follows:

Module	Description
`GetCentre()`	• Calculates the coordinates of the centre point • Uses the four modules: `FirstRowSet()`, `LastRowSet()`, `FirstColSet()`, `LastColSet()` • Assigns values to `CentreRow` and `CentreCol` • Sets `CentreRow` to −1 if no screen touch is detected

Write pseudocode for the module `GetCentre()`.

```
PROCEDURE GetCentre ()
    DECLARE StartRow, EndRow, StartCol, EndCol : INTEGER

    StartRow ← FirstRowSet()
    IF StartRow = -1 THEN
        CentreRow ← -1 // no 'touch' detected
    ELSE
        EndRow ← LastRowSet()
        StartCol ← FirstColSet()
        EndCol ← LastColSet()
        CentreRow ← INT((StartRow + EndRow)/2)
        CentreCol ← INT((StartCol + EndCol)/2)
    ENDIF
ENDPROCEDURE
```

..... [6]