Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

6785454555

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2022**

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 draws a program flowchart to show the sequence of steps required to solve a problem.

Give the technical term for a sequence of steps that describe how to solve a problem.

..... An algorithm .....

..... [1]

(b) The table lists some of the variables used in a program.

(i) Complete the table by writing the most appropriate data type for each variable.

Variable	Use of variable	Data type
`Temp`	Stores the average temperature	REAL
`PetName`	Stores the name of my pet	STRING
`MyDOB`	To calculate the number of days until my next birthday	DATE
`LightOn`	Stores state of light; light is only on or off	BOOLEAN

[4]

(ii) One of the names used for a variable in the table in part **1(b)(i)** is not an example of good practice.

Identify the variable and give a reason why it is **not** good practice to use that name.

Variable ... Temp .....

Reason .... Name does not indicate what the variable is used for .....

.....

.....

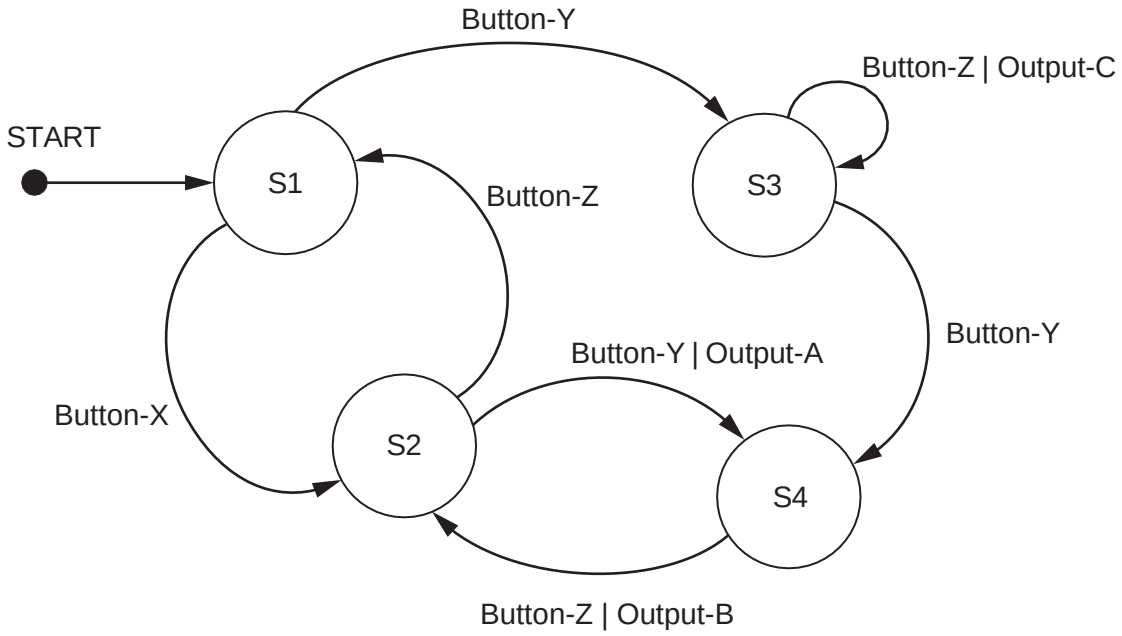
[2]

(c) Complete the table by evaluating each expression.

Expression	Evaluation
`INT((31 / 3) + 1)`	11
`MID(TO_UPPER("Version"), 4, 2)`	"SI"
`TRUE AND (NOT FALSE)`	TRUE
`NUM_TO_STR(27 MOD 3)`	"0"

[4]

2 Examine the following state-transition diagram.



(a) Complete the table with reference to the diagram.

	Answer
The number of different inputs	3
The number of different outputs	3
The single input value that could result in S4	Button-Y

[3]

(b) The initial state is S1.

Complete the table to show the inputs, outputs and next states.

Input	Output	Next state
Button-Y		
	none	
Button-Z		S2
	none	

[4]

Input	Output	Next state
Button-Y	none	S3
Button-Y	none	S4
Button-Z	Output-B	S2
Button-Z	none	S1

3 The manager of a cinema wants a program to allow users to book seats. The cinema has several screens. Each screen shows a different film.

(a) Decomposition will be used to break the problem down into sub-problems.

Describe **three** program modules that could be used in the design.

Module 1 .....

.....

.....

Module 2 ...

.....

.....

Module 3 .....

.....

.....

- Allows the user to search for films being shown // input name of film they want to see
- Allows the user to search for available seats
- Calculate cost of booking
- Book a given number of seats for a particular screening

[3]

(b) Two types of program modules may be used in the design of the program.

Identify the type of program module that should be used to return a value.

. ..... Function ..... [1]

BLANK PAGE

4 A stack is created using a high-level language. Memory locations 200 to 207 are to be used to store the stack.

The following diagram represents the current state of the stack.

`TopOfStack` points to the last value added to the stack.

Stack **Pointer**

Memory location	Value	
200		
201		
202		
203	'F'	← `TopOfStack`
204	'C'	
205	'D'	
206	'E'	
207	'H'	

(a) Complete the following table by writing the answers.

	Answer
The value that has been on the stack for the longest time.	'H'
The memory location pointed to by `TopOfStack` if **three** POP operations are performed.	206

[2]

(b) The following diagram shows the current state of the stack:

Stack **Pointer**

Memory location	Value	
200		
201		
202	'W'	← TopOfStack
203	'Y'	
204	'X'	
205	'Z'	
206	'N'	
207	'P'	

The following operations are performed:

```
POP
POP
PUSH 'A'
PUSH 'B'
POP
PUSH 'C'
PUSH 'D'
```

Complete the diagram to show the state of the stack **after** the operations have been performed.

Stack **Pointer**

Memory location	Value
200	
201	
202	
203	
204	
205	
206	
207	

Stack **Pointer**

Memory location	Value	
200		
201	'D'	← TopOfStack
202	'C'	
203	'A'	
204	'X'	
205	'Z'	
206	'N'	
207	'P'	

5 Each line of a text file contains data organised into fixed-length fields as shown:

<Field 1><Field 2><Field 3>

An algorithm is required to search for the first instance of a given value of Field 2 and, if found, to output the corresponding values for Field 1 and Field 3.

Describe the algorithm needed.

Do **not** include pseudocode statements in your answer.

1	Open file in read mode
2	Set up a conditional loop, repeating until the value is found or the `EOF()` is reached
3	Read a line from the file in a loop
4	Extract Field 2
5	Description of how Field 2 could be extracted e.g. using substring function and lengths of Field 1 and Field 2
6	Compare extracted field with search value
7	If search value found, extract Field 1and Field 3 and output them
8	Close the file after loop has finished

[6]

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

- output **each** integer value between 100 and 200 that ends with the digit 7, for example, 107
- output a final count of the number of values that are output.

Write pseudocode for this algorithm.

Any variables used must be declared.

```
DECLARE ThisInt, Count : INTEGER
Count ← 0

FOR ThisInt ← 100 TO 200
   IF ThisInt MOD 10 = 7 THEN
      OUTPUT ThisInt
      Count ← Count + 1
   ENDIF
NEXT ThisInt

OUTPUT Count
```

Alternative Solution:

```
DECLARE ThisInt, Count : INTEGER
Count ← 0

FOR ThisInt ← 107 TO 197 STEP 10
   OUTPUT ThisInt
   Count ← Count + 1
NEXT ThisInt

OUTPUT Count
```

[5]

(b) Study the following pseudocode.

```
CASE OF MySwitch
   1: ThisChar ← 'a'
   2: ThisChar ← 'y'
   3: ThisChar ← '7'
   OTHERWISE: ThisChar ← '*'
ENDCASE
```

Write pseudocode with the same functionality **without** using a `CASE` structure.

```
IF MySwitch = 1 THEN
   ThisChar ← 'a'
ELSE
   IF MySwitch = 2 THEN
      ThisChar ← 'y'
   ELSE
      IF MySwitch = 3 THEN
         ThisChar ← '7'
      ELSE
         ThisChar ← '*'
      ENDIF
   ENDIF
ENDIF
```

[4]

7 A string is a palindrome if it reads the same forwards as backwards.

The following strings are examples of palindromes:

"Racecar"
"madam"
"12344321"

Upper-case and lower-case characters need to be treated the same. For example, 'A' is equivalent to 'a'.

(a) A function `IsPalindrome()` will take a string parameter. The function will return `TRUE` if the string is a palindrome and will return `FALSE` if the string is not a palindrome.

Write pseudocode for `IsPalindrome()`.

```
FUNCTION IsPalindrome(InString : STRING) RETURNS BOOLEAN
  DECLARE IsPal : BOOLEAN
  DECLARE Index, Num : INTEGER
  DECLARE CharA, CharB : CHAR

  IsPal ← TRUE
  Index ← 1

  Num ← INT(LENGTH(InString) / 2)

  WHILE Index <= Num AND IsPal = TRUE
     CharA ← MID(InString, Index, 1)
     CharB ← MID(Instring, LENGTH(Instring) - Index + 1,
                 1)
     IF UCASE(CharA) <> UCASE(CharB) THEN
        IsPal ← FALSE // RETURN FALSE
     ENDIF
     Index ← Index + 1
  ENDWHILE

  RETURN IsPal // RETURN TRUE

ENDFUNCTION
```

.....

.....

.....

.....

.....

.....

.....

.....

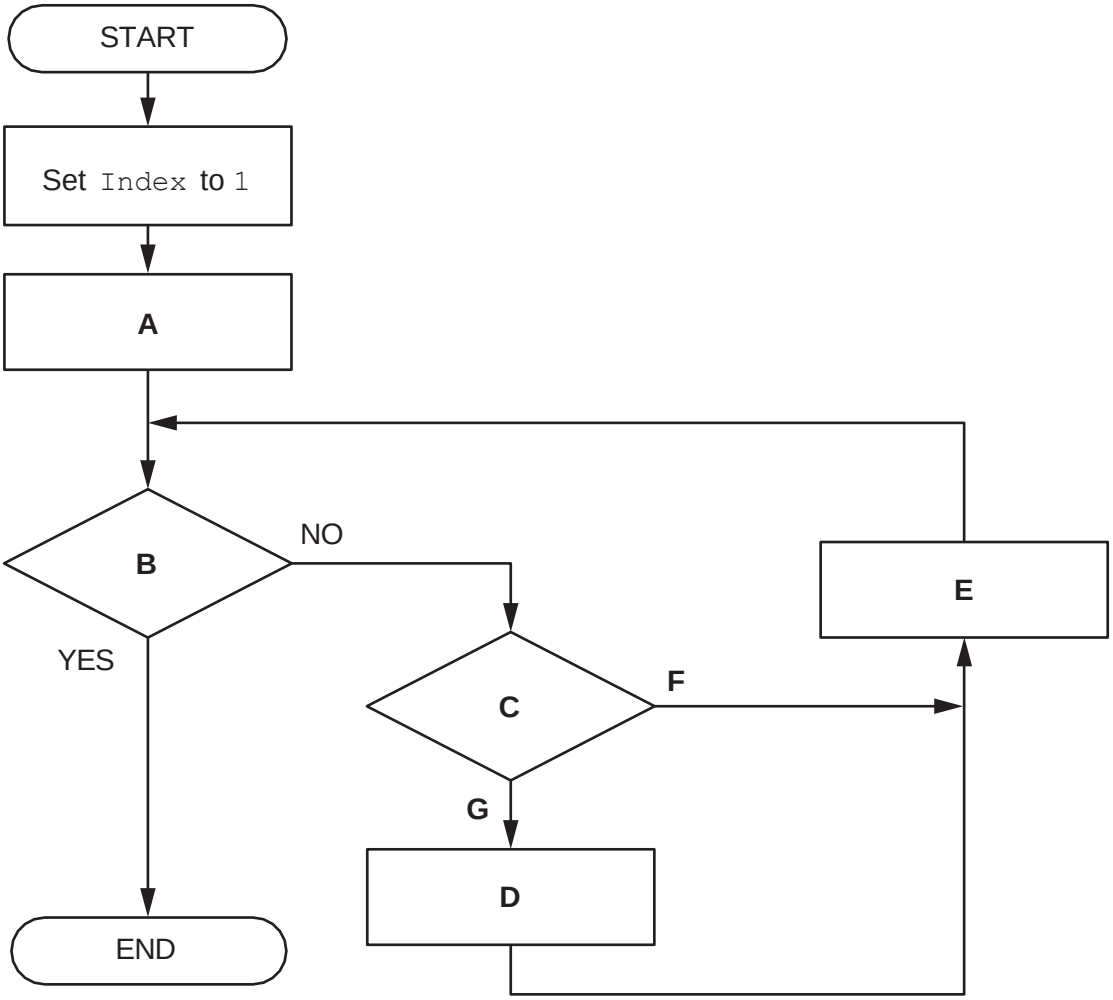
.....

..... [7]

(b) Strings may consist of several words separated by spaces.

For example, the string "never odd or even" becomes a palindrome if the spaces are removed.

The program flowchart represents an algorithm to produce a string `OutString` by removing all spaces from a string `InString`.



Complete the table by writing the text that should replace each of the labels **B**, **C**, **D**, **F** and **G**.

Note: the text may be written as a pseudocode statement.

Label	Text
A	Set `OutString` to `""`
B	
C	
D	
E	Set `Index` to `Index + 1`
F	
G	

A	Set `OutString` to `""`
B	Is **`Index > LENGTH(InString)`**?
C	Is **`MID(InString, Index, 1) = " "`**?
D	Set **`OutString`** to **`OutString & MID(InString, Index, 1)`**
E	Set **`Index`** to **`Index + 1`**
F	**YES**
G	**NO**

[4]

8 A program allows a user to save passwords used to login to websites. A stored password is inserted automatically when the user logs into the corresponding website.

A student is developing a program to generate a password. The password will be of a fixed format, consisting of **three groups of four** alphanumeric characters. The groups are separated by the hyphen character '-'.

An example of a password is: "FxAf-3haV-Tq49"

A global 2D array `Secret` of type `STRING` stores the passwords together with the website domain name where they are used. `Secret` contains 1000 elements organised as 500 rows by 2 columns.

Unused elements contain the empty string (`""`). These may occur anywhere in the array.

An example of a part of the array is:

Array element	Value
`Secret[27, 1]`	"www.thiswebsite.com"
`Secret[27, 2]`	"••••••••••••••"
`Secret[28, 1]`	"www.thatwebsite.com"
`Secret[28, 2]`	"••••••••••••••"

Note:
- For security, passwords are stored in an encrypted form, shown as "••••••••••••••" in the example.
- The passwords cannot be used without being decrypted.
- Assume that the encrypted form of a password will **not** be an empty string.

The programmer has started to define program modules as follows:

Module	Description

`RandomChar()`	• Generates a single random character from within one of the following ranges:
	  ○ 'a' to 'z'
	  ○ 'A' to 'Z'
	  ○ '0' to '9'
	• Returns the character

`Encrypt()`	• Takes a password as a parameter of type string • Returns the encrypted form of the password as a string
`Decrypt()`	• Takes an encrypted password as a parameter of type string • Returns the decrypted form of the password as a string

For reference, relevant ASCII values are as follows:

Character range	ASCII range
'a' to 'z'	97 to 122
'A' to 'Z'	65 to 90
'0' to '9'	48 to 57

(a) Write pseudocode for module `RandomChar()`.

You may wish to refer to the **insert** for a description of the `CHR()` function. Other functions may also be required.

```
FUNCTION RandomChar() RETURNS CHAR
  DECLARE ThisRange : INTEGER
  DECLARE ThisChar : CHAR

  //First select the range
  ThisRange ← INT(RAND(3)) + 1 // 1 to 3

  CASE OF ThisRange
    1: ThisChar ← CHR(INT(RAND(26) + 65)) // 65 to 90:
                                          'A' to 'Z'
       ThisChar ← LCASE(ThisChar)       // 'a' to 'z'
    2: ThisChar ← CHR(INT(RAND(26) + 65)) // 65 to 90:
                                          A to Z
    3: ThisChar ← NUM_TO_STR(INT(RAND(10)) // '0' to '9'
  ENDCASE

  RETURN ThisChar
ENDFUNCTION
```

.....

.....

.....

.....

.....

.....

.....

..... [6]

(b) A new module is defined as follows:

Module	Description
`FindPassword()`	• Takes a website domain name as a parameter of type string • Searches for the website domain name in the array `Secret` • If the website domain name is found, the decrypted password is returned • If the website domain name is not found, a warning message is output, and an empty string is returned

Write pseudocode for module `FindPassword()`.

Assume that modules `Encrypt()` and `Decrypt()` have already been written.

```
FUNCTION FindPassword(Name: STRING) RETURNS STRING
  DECLARE Index : INTEGER
  DECLARE Password : STRING

  Password ← ""
  Index ← 1

  WHILE Password = "" AND Index <= 500
     IF Secret[Index, 1] = Name THEN
       Password ← Decrypt(Secret[Index, 2])
     ELSE
       Index ← Index + 1
     ENDIF
  ENDWHILE

  IF Password = "" THEN
     OUTPUT "Domain name not found"
  ENDIF

  RETURN Password

ENDFUNCTION
```

[7]

(c) The modules `Encrypt()` and `Decrypt()` are called from several places in the main program.

Identify a method that could have been used to test the main program before these modules were completed. Describe how this would work.

Method ..... Stub testing .....

Description ....

- A simple module is written to replace each of the modules.
- The simple module will return an expected value // will output a message to show they have been called

..... [3]

(d) A validation function is written to check that the passwords generated are valid.

To be valid, each password must:

- be 14 characters long
- be organised as three groups of four case-sensitive alphanumeric characters. The groups are separated by hyphen characters
- not include any duplicated characters, except for the hyphen characters.

Note: lower-case and upper-case characters are not the same. For example, 'a' is not the same as 'A'.

Give **two** password strings that could be used to test different areas of the validation rules.

Password 1 .....

Password 2 .....

One mark for each password example that breaks **one** of the rules due to:

- Length too long // length too short
- Invalid character
- Incorrect grouping (including number of hyphens)
- Duplicated characters

[2]

(e) The `RandomChar()` module is to be modified so that alphabetic characters are generated twice as often as numeric characters.

Describe how this might be achieved.

....

- Generate a random integer divisible by 3
- Split range into 1/3 and set as numeric
- Else alphabetic character

..... [3]

BLANK PAGE

BLANK PAGE

BLANK PAGE