Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME									
CENTRE NUMBER						CANDIDATE NUMBER			

7911800738

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2016**

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.
You may use an HB pencil for any diagrams, graphs or rough working.
Do not use staples, paper clips, glue or correction fluid.
DO **NOT** WRITE IN ANY BARCODES.

Answer **all** questions.
No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.
The number of marks is given in brackets [ ] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

DC (LK/SG) 106693/4

CAMBRIDGE International Examinations

[Turn over

There is an **Appendix** on the last page. Some questions will refer you to this information.

1 The items in the table below are statements from a program in a generic programming language.

For the built-in functions list, refer to the **Appendix** on the last page.

(a) (i) Show what type of programming construct each statement represents.

Complete the table by putting a tick (✓) in the appropriate column for each item.

Item	Statement	Selection	Iteration	Assignment
1	`WHILE DegF > 37.5`			
2	`MyName = "Gordon"`			
3	`DegF = INT(DegF)`			
4	`ENDIF`			
5	`CASE OF MyFavourite`			
6	`UNTIL x = 5`			

[6]

(ii) State the purpose of each statement in the table in **part (a)(i)**.

Do **not** use mathematical symbols in your descriptions.

Item	Purpose of statement
1	....
2	....
3	....
4	....
5	....
6	....

[6]

(iii) Evaluate the following expressions when `MyString` has the value `"Corrective Maintenance"`:

Expression	Evaluates to
`'P' & MID(MyString, 13, 4)`	
`RIGHT(MID(MyString, 6, 10) ,4)`	

[2]

2 The engine management system of a car includes an energy-saving facility. When certain conditions are met, this facility will automatically stop the engine.

The system is to be software-based. It will include a subroutine, `EnergySaver`, which repeatedly takes data from sensors in the car. The subroutine decides whether or not to set the `EngineStop` value.

The table of identifiers used by this subroutine is shown below.

(a) Complete the identifier table below by stating the data types.

Identifier	Data type	Description
`Accelerator`	.....	Accelerator pedal position Values: 0 to 100 in steps of 1 Meaning: 0: none (not pressed) 100: maximum (fully pressed)
`EngineTemp`	.....	Engine temperature in °C (-50 to +150 stored to 1 decimal place)
`NormalTemp`	.....	Normal engine temperature in °C Whole number; typical value 90
`Speed`	.....	Road speed of car (in km/hr) Values: 0 to 200 in steps of 1
`EngineStop`	.....	Value used to signal engine must be stopped Possible values: TRUE: stop engine FALSE: run engine

[5]

The condition for stopping the engine is that all three of the following are true:

- Accelerator is not pressed
- Engine temperature is normal or above
- Car speed is zero

The initial design stage will produce a prototype of `EnergySaver`, with a user interface.

The structured English for this is:

1. INPUT value for accelerator pedal position
2. INPUT value for engine temperature
3. INPUT value for normal engine temperature
4. INPUT value for car speed
5. EVALUATE engine stopping condition
6. IF stopping condition satisfied SET engine stop value to TRUE
7. IF stopping condition not satisfied SET engine stop value to FALSE
8. OUTPUT message indicating engine stop value

(b) Write the **pseudocode** equivalent of the structured English. Use the identifiers from the table in **part (a)**.

For the built-in functions list, refer to the **Appendix** on the last page.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

3 String encryption was implemented using a simple character-substitution method. A function, `Decrypt`, is needed to reverse the encryption process and return the original character.

The encryption uses the 7-bit ASCII value for each character. This value is used as an index for a 1D array, `Lookup`, which contains the substitute characters.
`Lookup` contains an entry for each of the ASCII characters.

This function, `Decrypt`, will accept two parameters, a single character, `CipherChar`, and the 1D array, `Lookup`.

The steps involved in `Decrypt` are follows:

- Search for the character in the array
- Note the index value where the character is found (the index value is the ASCII value of the original character)
- Use the index value to obtain the original character

(a) The first attempt at writing the pseudocode for this function is shown below.

Complete the pseudocode.

For the built-in functions list, refer to the **Appendix** on the last page.

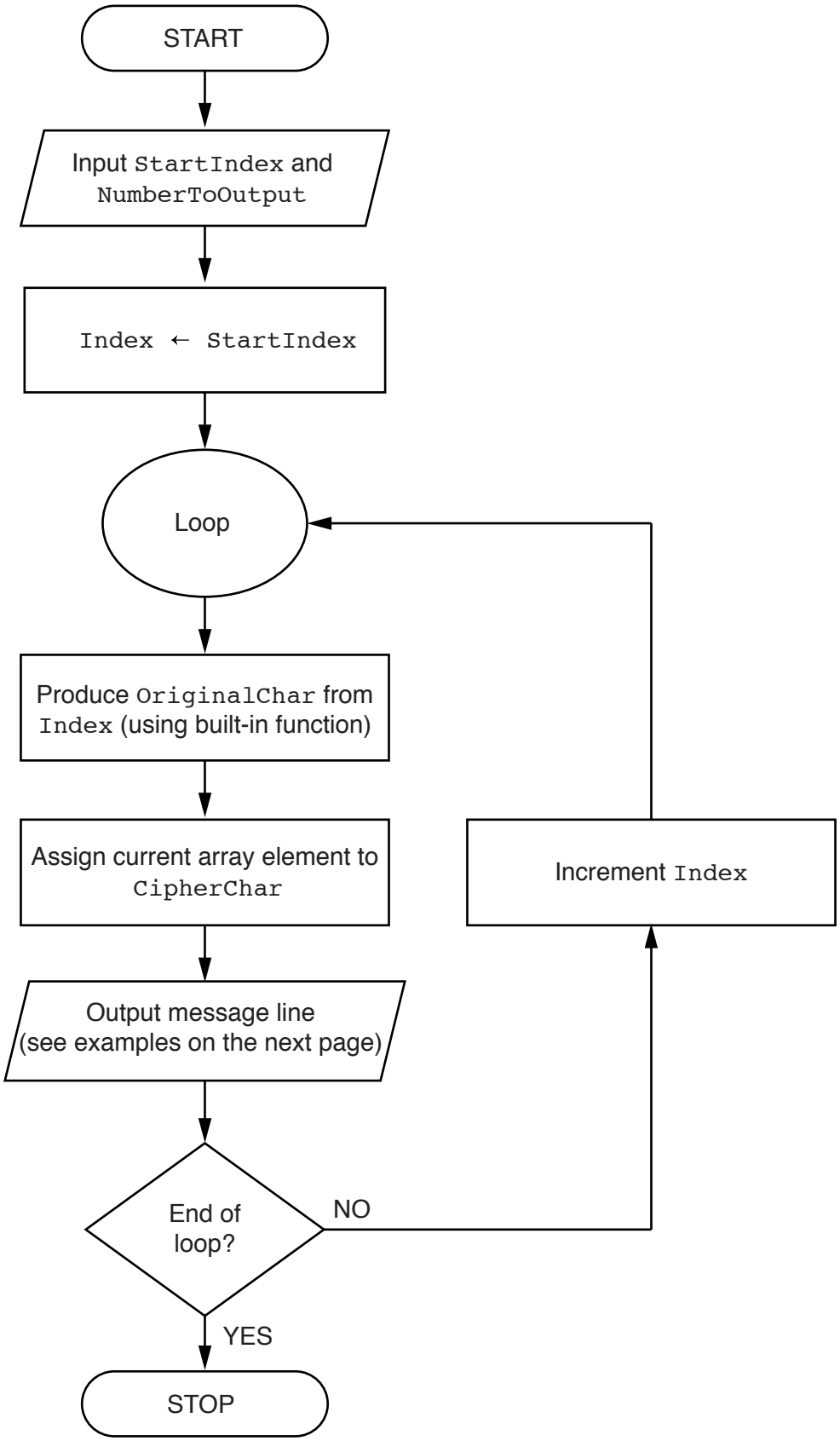
```
FUNCTION Decrypt(..... , .....) RETURNS CHAR

   DECLARE Found     : .....

   DECLARE ..... : INTEGER

   DECLARE OriginalChar     : CHAR

   Index ← 1        // .....

   Found ← FALSE
   //search for CipherChar in Lookup:

   WHILE .....
      //compare CipherChar with this array element:

      IF .....
         THEN
            .....//Set the flag
         ELSE
            Index .....//Move to next array element
      ENDIF
   ENDWHILE
   //dropped out of loop so must have found CipherChar:

   .....//convert Index to original character

   RETURN .....

ENDFUNCTION
```

[11]

Question 3 continues on page 8.

(b) A program is to be written to output part of the `Lookup` array.

The design of the algorithm is shown below.
It may be assumed that the characters output from `Lookup` are all printable.



For example, for the input of 65 and 3, the output will be:

```
Index 65: Character A has substitute character Y
Index 66: Character B has substitute character Q
Index 67: Character C has substitute character F
```

Write **program code** to implement the flowchart design.

In addition to the `Lookup` array, assume that the following variables have been declared:

`StartIndex, NumberToOutput, Index`

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

[6]

4 **(a)** Name **two** features of your chosen high-level programming language that support the implementation of a modular design.

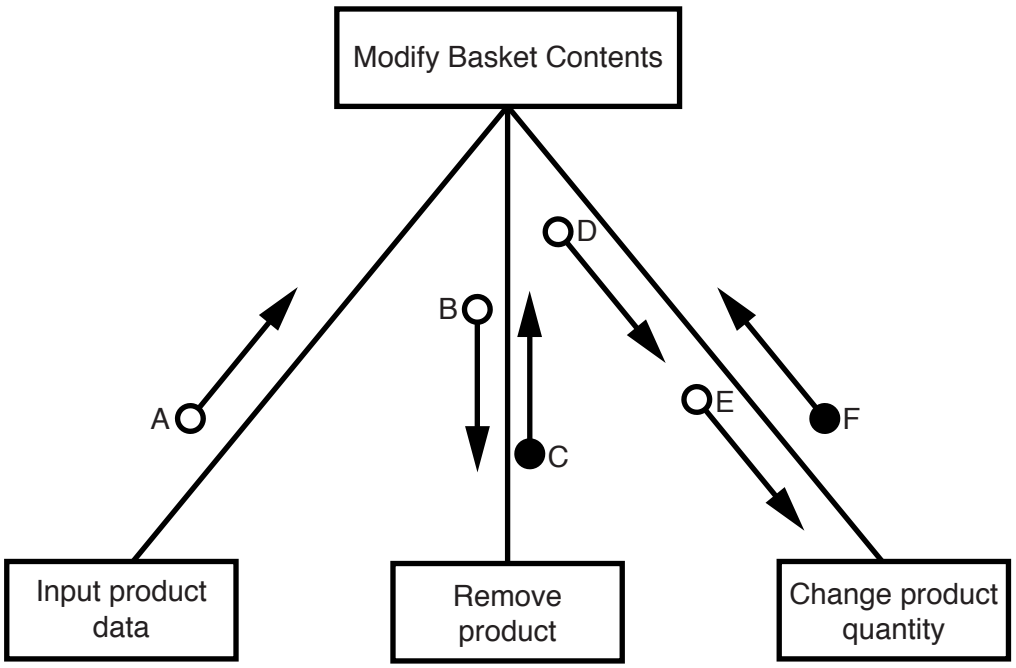
1 .....

2 .....
[2]

(b) **(i)** The structure chart shows part of the design of a program for an online shopping system.

The user has already added a number of products to their virtual basket.

Draw on the chart, the symbol to show that the process of modifying the basket contents may be iterated (repeated).



[1]

(ii) Each arrow in the structure chart above represents a parameter.

The table below shows the three data items that the six parameters pass between modules.

Tick (✓) to match each parameter to the correct data item.

Data item	Parameter					
	A	**B**	**C**	**D**	**E**	**F**
Product ID						
Quantity						
Flag Value – indicating operation success or fail						

[4]

5 Claudia stores her large collection of music CDs in different places. Claudia wants to record where she stores each CD. She decides to write a program to do this.

Data items for a typical CD are:

Title: Kind of Green
Artist: Miles Coltrane
Location: Rack3-23

The data is to be stored in a text file, `MyMusic`. Each line of the text file will be a string, formed by concatenating the three data items.

Before concatenation, the title and artist will each be made into a fixed-length string of 40 characters. Space characters may need to be added to each data item.

The location is always 8 characters long.

(a) (i) Explain the benefit of making the stored data into fixed-length strings.

.....

.....

.....

.....

State a drawback of this file design.

.....

.....
[3]

(ii) When Claudia buys a new CD, the CD data must be added to the existing file, `MyMusic`. She has written a procedure in pseudocode. This has the following file-handling statements:

```
OPENFILE "MyMusic" FOR WRITE
WRITEFILE "MyMusic", OutputString
CLOSEFILE "MyMusic"
```

There is a problem with the logic of this pseudocode.

State the problem.

.....

.....

Identify the effect it will have if the final code is implemented in this way.

.....

.....

Give a possible solution.

.....

.....

[3]

(b) Claudia needs to output a list of all the CDs in a particular location.

She designs a procedure, `OutputLocationList`, to do this. She also chooses the following identifiers:

Identifier	Data type
CDTitle	STRING
CDArtist	STRING
CDLocation	STRING

The procedure will:

- prompt for the name of the location
- input the location (such as “Rack3-23”)
- search the file for all CDs at this location
- output the title and artist of each CD found
- output the total number of CDs found at that location (such as “17 CDs found”)

Write **program code** for the procedure `OutputLocationList`.

Visual Basic and Pascal: You should include the declaration statements for variables.
Python: You should show a comment statement for each variable used with its data type.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

[10]

6 A string-handling function has been developed.

For the built-in functions list, refer to the **Appendix** on the last page.

The pseudocode for this function is shown below.

```
FUNCTION SF(ThisString : STRING) RETURNS STRING
   DECLARE x          : CHAR
   DECLARE NewString  : STRING
   DECLARE Flag       : BOOLEAN
   DECLARE m, n       : INTEGER

   Flag ← TRUE
   NewString ← ""
   m ← LENGTH(ThisString)

   FOR n ← 1 TO m

      IF Flag = TRUE
         THEN
            x ← UCASE(MID(ThisString, n, 1))
            Flag ← FALSE
         ELSE
            x ← LCASE(MID(ThisString, n, 1))
      ENDIF

      NewString ← NewString & x

      IF x = " "
         THEN
            Flag ← TRUE
      ENDIF

   ENDFOR

   RETURN NewString
ENDFUNCTION
```

(a) (i) Complete the trace table below by performing a dry run of the function when it is called as follows:

```
SF("big BEN")
```

n	x	Flag	m	NewString

[4]

(ii) Describe the purpose of function `SF`.

.....

.....

.....

.....[2]

(b) Test data must be designed for the function `SF`.

(i) State what happens when the function is called with an empty string.

.....

.....[1]

(ii) The function should be thoroughly tested.

Give **three** examples of non-empty strings that may be used.

In each case explain why the test string has been chosen.

String .....

Explanation .....

.....

String .....

Explanation .....

.....

String .....

Explanation .....

.....

[3]

Appendix

Built-in functions (Pseudocode)

In each function below, if the function call is not properly formed, the function returns an error.

```
MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING
```

returns the string of length `y` starting at position `x` from `ThisString`
Example: **`MID("ABCDEFGH", 2, 3)`** will return string **`"BCD"`**

```
LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING
```

returns the leftmost `x` characters from `ThisString`
Example: **`LEFT("ABCDEFGH", 3)`** will return string **`"ABC"`**

```
RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING
```

returns the rightmost `x` characters from `ThisString`
Example: **`RIGHT("ABCDEFGH", 3)`** will return string **`"FGH"`**

```
CHR(x : INTEGER) RETURNS CHAR
```

returns the character whose ASCII value is `x`
Example: **`CHR(87)`** will return **`'W'`**

```
ASC(x : CHAR) RETURNS INTEGER
```

returns the ASCII value of character `x`
Example: **`ASC('W')`** will return **`87`**

```
LCASE(x : CHAR) RETURNS CHAR
```

returns the lower case equivalent character of `x`
Example: **`LCASE('W')`** will return **`'w'`**

```
UCASE(x : CHAR) RETURNS CHAR
```

returns the upper case equivalent character of `x`
Example: **`UCASE('h')`** will return **`'H'`**

```
INT(x : REAL) RETURNS INTEGER
```

returns the integer part of `x`
Example: **`INT(27.5415)`** will return **`27`**