Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

9930793408

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 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 **20** printed pages.

CAMBRIDGE International Examinations

There is an **Appendix** on pages 19 and 20. Some questions will refer you to this information.

1 A programmer wants to write a program to calculate the baggage charge for a passenger's airline flight.

Two types of ticket are available for a flight:

- economy class (coded E)
- standard class (coded S)

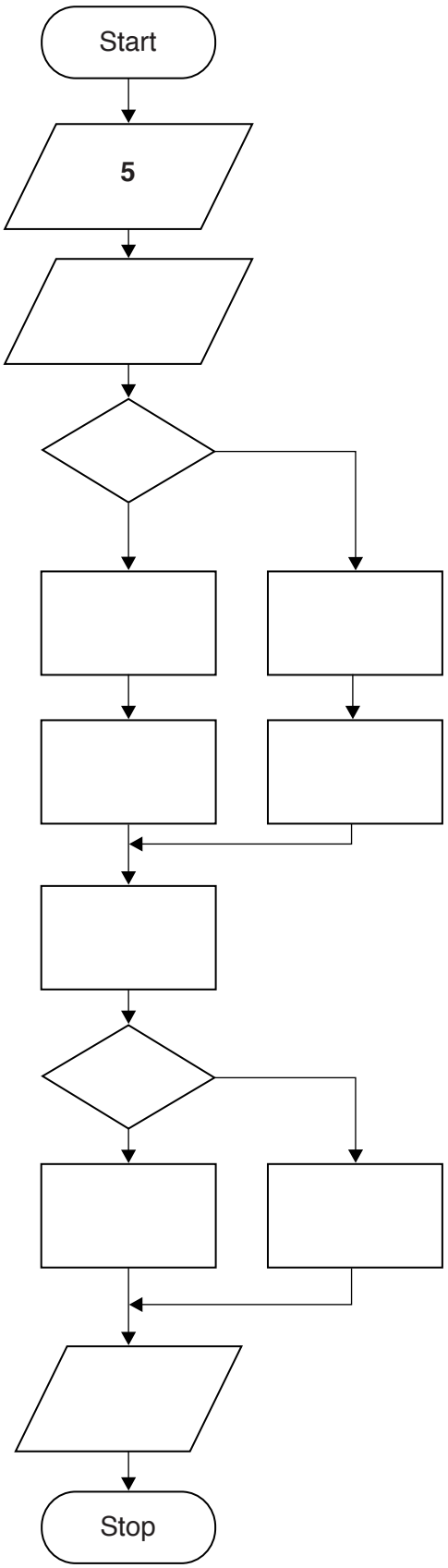
Each ticket type has a baggage weight allowance as shown below. The airline makes a charge if the weight exceeds the allowance.

Ticket type	Baggage allowance (kg)	Charge rate per additional kg ($)
'E'	16	3.50
'S'	20	5.75

(a) A program flowchart will document the program. The flowchart will contain the following statements:

Statement number	Statement
1	`Charge ← 0`
2	`INPUT BaggageWeight`
3	`Charge ← ExcessWeight * ChargeRate`
4	`Is ExcessWeight > 0 ?`
5	`INPUT TicketType`
6	`ExcessWeight ← BaggageWeight - BaggageAllowance`
7	`BaggageAllowance ← 16`
8	`ChargeRate ← 3.5`
9	`OUTPUT Charge`
10	`ChargeRate ← 5.75`
11	`BaggageAllowance ← 20`
12	`Is TicketType = 'E' ?`

Complete the flowchart by putting the appropriate **statement number** in each flowchart symbol. Statement 5 has been done for you.



[6]

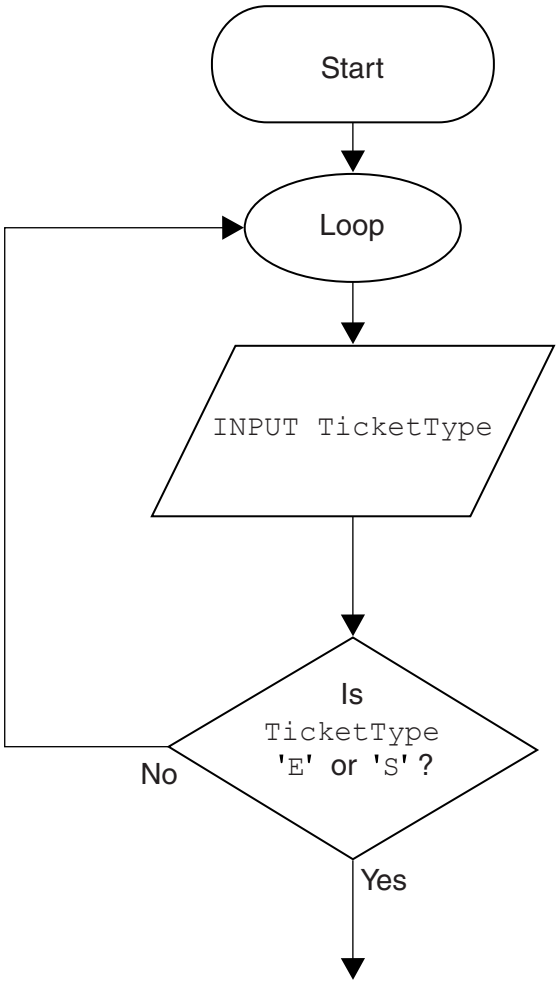
(b) The programmer needs data to test the flowchart.

Complete the table of test data below to show **five** tests.

`TicketType`	`BaggageWeight`	Explanation	Expected output
E	15	....	
		....	
		....	
		....	
		....	

[5]

(c) The program design is to be amended. The value input by the user for the ticket type is to be validated. Part of the amended flowchart is shown below.



Write **pseudocode** to use a pre-condition loop for this validation.

.....

.....

.....

.....

.....

.....[3]

2 A sensing device sends bit values to a computer along data channels.

- Channel 1 transmits a sequence of binary values from a sensor
- Channel 2 transmits at regular intervals to indicate whether the sensor is switched on or off:
  - 0 indicates switched off
  - 1 indicates switched on

A program tests the bits received from the sensing device.

A program reads the signal from Channel 2 after every six values from Channel 1.

A built-in function `READ(<ChannelNumber>)` reads a value from the specified channel.

Pseudocode for the program is as follows:

```
01      BitCount ← 0
02      Status2 ← READ(2)
03      WHILE Status2 = 1
04
05        FOR ReadingCount ← 1 TO 6
06          ThisBit ← READ(1)
07          IF ThisBit = 1
08            THEN
09              BitCount ← BitCount + 1
10          ENDIF
11          IF BitCount = 5
12            THEN
13              OUTPUT "Error – Investigate"
14              BitCount ← 0
15          ENDIF
16        ENDFOR
17
18        Status2 ← READ(2)
19      ENDWHILE
```

(a) Trace the execution of the program for the following sequence of bits.

Channel 1		1	0	1	1	1	0		1	1	0	0	1	1	
Channel 2	1							1							0

Status2	ReadingCount	ThisBit	BitCount	OUTPUT
			0	
1	1	1	1	
	2			

[7]

(b) Identify the following constructs in the given program, using line numbers.

For multi-line constructs give the first line number only.

Construct	Line number
Assignment	
Selection	
Iteration	

[3]

3 You will need to refer to the list of pseudocode string-handling functions in the **Appendix**.

ASCII code table (part)					
Character	Decimal	Character	Decimal	Character	Decimal
<Space>	32	I	73	R	82
A	65	J	74	S	83
B	66	K	75	T	84
C	67	L	76	U	85
D	68	M	77	V	86
E	69	N	78	W	87
F	70	O	79	X	88
G	71	P	80	Y	89
H	72	Q	81	Z	90

(a) For each statement, write the value assigned to the variable.

(i) `Term ← CHARACTERCOUNT("TSUNAMI")`

`Term` .....[1]

(ii) `Answer1 ← ASC('G') + ASC(<Space>)`

`Answer1` .....[1]

(iii) `Answer2 ← CHR(CHARACTERCOUNT("HELLO") + 70)`

`Answer2` .....[1]

(iv) `Word ← SUBSTR("Welcome home", 4, 7) )`

`Word` .....[1]

Question 3(b) continues on page 10.

(b) A programmer wants to design a procedure to calculate a customer ID number from the customer's surname.

The procedure will:

- input the surname
- isolate each character in the surname and find the corresponding ASCII code
- calculate the total of all these ASCII codes
- this total is the customer ID

(i) Complete the pseudocode for this procedure.

You will need to refer to the list of pseudocode string-handling functions in the Appendix.

```
PROCEDURE CalculateCustomerID

  OUTPUT "Key in surname"

  INPUT Surname

  Length ← .....

  CustomerID ← 0

  FOR i ← 1 TO Length

    // NextChar is a single character from Surname

    NextChar ← .....

    NextCodeNumber ← ASC(NextChar)

    CustomerID ← CustomerID + .....

  ENDFOR

  OUTPUT "Customer ID is ", CustomerID
```

[3]

(ii) Write **program code** for procedure `CalculateCustomerID`.

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

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

(c) The programmer decides that it would be better to write the procedure as a function. The user will now input the surname in the main program.

Write **program code** for the following:

State your programming language .....

(i) The function header for this new function `CalculateCustomerID`

.....[3]

(ii) The additional statement required within the function body to complete the change from a procedure to a function.

.....

.....[1]

(iii) The statement in the main program which:

- calls the function for surname Wilkes
- assigns the result to variable `ThisID`

.....[3]

(d) (i) The new function `CalculateUserID` is an example of a 'user-defined function'.

State **two** differences between a built-in function and a user-defined function.

1 .....

.....

2 .....

.....[2]

(ii) State **two** things that built-in and user-defined functions have in common.

1 .....

.....

2 .....

.....[2]

4 A company employs Ahmed as a programmer.

(a) At College, before joining the company, Ahmed used two items of software for programming:

- a text editor
- a compiler

Describe how he could have developed programs using these software tools.

Include in the description the terms 'object code' and 'source code'.

.....

.....

.....

.....

.....

.....

.....[3]

(b) Ahmed now uses an Integrated Development Environment (IDE) for programming.

(i) State **one** feature an IDE provides to help with the identification of syntax errors.

.....

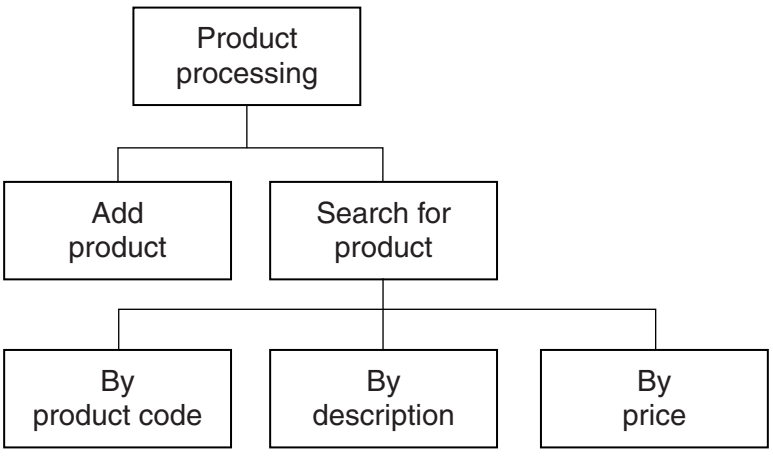
.....[1]

(ii) State **one** feature an IDE provides to carry out white box testing.

.....

.....[1]

(c) The company maintains a file of product data. Ahmed is to write a program to add a new product and search for a product based on the structure diagram shown:



The program records the following data for each product:

- product code
- product description
- product retail price

The text file `PRODUCTS` stores each data item on a separate line, as shown below:

File **PRODUCTS**

0198
Plums(10kg)
11.50
0202
Onions(20kg)
10.00
~
0376
Mango chutney(1kg)
02.99
~
0014
Mango(10kg)
12.75

The program uses the variables shown in the identifier table.

Identifier	Data type	Description
PRODUCTS	TEXT FILE	Storing the code, description and retail price for all current products
PCode	ARRAY[1:1000] OF STRING	Array storing the product codes
PDescription	ARRAY[1:1000] OF STRING	Array storing the product descriptions
PRetailPrice	ARRAY[1:1000] OF REAL	Array storing the product retail prices
i	INTEGER	Array index used by all three arrays

(i) The first operation of the program is to read all the product data held in file PRODUCTS and write them into the three 1D arrays.

Complete the pseudocode below.

```
OPEN .....

i ← 1

WHILE .....

  READFILE ("PRODUCTS", .....)

  READFILE ("PRODUCTS", .....)

  READFILE ("PRODUCTS", .....)

  .....

  .....

ENDWHILE

CLOSE "PRODUCTS"

OUTPUT "Product file contents written to arrays"
```

[5]

When Ahmed designed the PRODUCTS file, he considered the alternative file structure shown opposite.

It stores one product per line in the text file.

File **PRODUCTS**

0198 Plums(10kg)	11.50
0202 Onions(20kg)	10.00
~	
0376 Mango chutney(1kg)	02.99
~	
0014 Mango(10kg)	12.75

(ii) State **one** benefit and **one** drawback of this file design.

Benefit .....

.....

Drawback .....

.....[2]

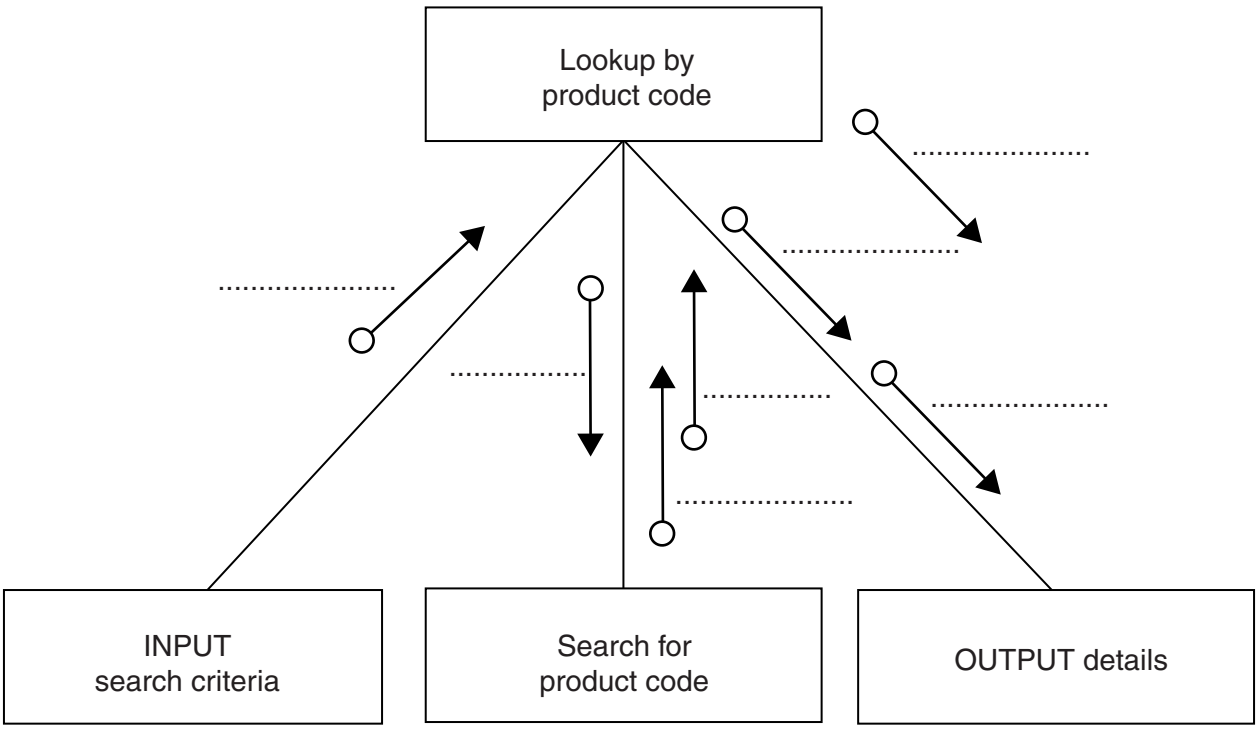
(d) To code the 'Search by product code' procedure, Ahmed draws a structure chart showing the different stages.

The procedure uses the variables shown in the identifier table.

Identifier	Data type	Description
`SearchCode`	`STRING`	Product code input by the user
`ThisIndex`	`INTEGER`	Array index position for the corresponding product
`ThisDescription`	`STRING`	Product description found
`ThisRetailPrice`	`REAL`	Product retail price found

You can assume that before the procedure is run, all the product data is read from file `PRODUCTS` and then stored in three 1D arrays as described in **part (c)(i)**.

Label the structure chart to show the input(s) and output(s).



[4]

(e) A first attempt was made at writing the 'Search for product code' module.
Ahmed designs this as a function `ProductCodeSearch`.

The function returns an integer value as follows:

- if the product code is found, it returns the index position of the 1D array `PCode` being searched
- if the product code is not found, the function returns -1

Write **program code** for function `ProductCodeSearch`.

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[6]

5 Study the following pseudocode statements.

```
CONST  Pi =  3.1          : REAL
DECLARE Triangle, Base, Height, Radius, Cone : REAL
DECLARE a, b, c, Answer2 : INTEGER
DECLARE Answer1          : BOOLEAN
Base ← 2.6
Height ← 10
Triangle ← (Base * Height) / 2
Radius ← 1
Height ← 2
Cone  ← 2 * Pi * Radius * (Radius + Height)
a ← 13
b ← 7
c ← 3

Answer1 ← NOT((a + b + c) > 28)
Total ← 34
Total ← Total - 2
Answer2 ← a + c * c
```

Give the final value assigned to each variable.

(i) `Triangle` ..... [1]

(ii) `Cone` ..... [1]

(iii) `Answer1` ..... [1]

(iv) `Total` ..... [1]

(v) `Answer2` ..... [1]

Appendix

Built-in functions (pseudocode)

```
ONECHAR(ThisString : STRING, Position : INTEGER) RETURNS CHAR
```

returns the single character at position `Position` (counting from the start of the string with value 1) from the string `ThisString`.

For example: `ONECHAR("New York", 5)` returns `'Y'`

```
CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER
```

returns the number of characters in `ThisString`.

For example: `CHARACTERCOUNT("New York")` returns `8`

```
SUBSTR(ThisString : STRING, Value1 : INTEGER, Value2 : INTEGER) RETURNS
STRING
```

returns a sub-string from within `ThisString`.
`Value1` is the start index position (counting from the left, starting with 1).
`Value2` is the final index position.

For example: `SUBSTR("art nouveau", 5, 11)` returns `"nouveau"`

```
TONUM(ThisString : STRING) RETURNS INTEGER or REAL
```

returns the integer or real equivalent of the string `ThisString`.

For example: `TONUM("502")` returns the integer `502`
`TONUM("56.36")` returns the real number `56.36`

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

returns an integer which is the ASCII character code for the character `ThisCharacter`.

For example: `ASC('A')` returns integer `65`

`CHR(Value : INTEGER) RETURNS CHAR`

returns the character that ASCII code number `Value` represents.

For example: `CHR(65)` returns `'A'`

`RND() RETURNS REAL`

returns a random number in the range 0 to 0.99999

For example: `RND()` returns `0.67351`

`INT(ThisNumber : REAL) RETURNS INTEGER`

returns the integer part of `ThisNumber`.

For example: `INT(12.79)` returns `12`

Errors

For any function, if the program calls the function incorrectly, the function returns an error.

Concatenation operator

`&` – Concatenates two expressions of `STRING` or `CHAR` data type.

For example: `"South" & " " & "Pole"` produces `"South Pole"`
`'B' & "000654"` produces `"B000654"`