Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	
CENTRE NUMBER	
CANDIDATE NUMBER	

4630108004

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2015**

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.

CAMBRIDGE International Examinations

Throughout the paper you will be asked to write either **pseudocode** or **program code**.

Complete the statement to indicate which high-level programming language you will use.

Programming language .....

1 Computer programs have to evaluate expressions.

Study the sequence of pseudocode statements.

Give the value assigned to each variable.

The statement may generate an error. If so, write ERROR.

The & operator is used to concatenate strings.

```
DECLARE N1        : INTEGER
DECLARE N2        : INTEGER
DECLARE Answer    : REAL
DECLARE Found     : BOOLEAN
DECLARE IsValid   : BOOLEAN
N1 ← 3
N2 ← 9
Answer ← (N1 + N2) / 6
Answer ← 3 * (N1 - 2) + N2 / 2
IsValid ← (N1 > N2) AND (N2 = 9)
Found ← FALSE
IsValid ← (N1 > N2 / 2) OR (Found = FALSE)
Answer ← "1034" & " + " & "65"
```

(i) Answer ..... [1]

(ii) Answer ..... [1]

(iii) IsValid ..... [1]

(iv) IsValid ..... [1]

(v) Answer ..... [1]

2 A program is to simulate the operation of a particular type of logic gate.

- The gate has two inputs (0 or 1) which are entered by the user.
- The program will display the output (0 or 1) from the gate.

The program uses the following identifiers in the pseudocode below:

Identifier	Data type	Description
P	INTEGER	Input signal
Q	INTEGER	Input signal
X	INTEGER	Output signal

```
01  INPUT P
02  INPUT Q
03  IF (P = 1 AND Q = 0) OR (P = 0 AND Q = 1) OR (P = 0 AND Q = 0)
04     THEN
05        X ← 0
06     ELSE
07        X ← 1
08  ENDIF
09  OUTPUT X
```

(a) The programmer chooses the following four test cases.

Show the output (X) for each test case.

Test case	Input		Output
	P	Q	X
1	1	1	
2	1	0	
3	0	1	
4	0	0	

[4]

(b) The selection statement (lines 03 – 08) could have been written with more simplified logic.

Rewrite this section of the algorithm in **pseudocode**.

.....

.....

.....

.....

.....

.....[3]

3 Regular customers at a supermarket use a rewards card at the point-of-sale.

Points are calculated from every transaction and added to the points total stored on the card.

One reward point is given for every $1 spent.

When the points total exceeds 500, the customer can either:

- pay the full amount due and increase their points total
- get $1 deducted from the amount due in exchange for 500 reward points

The new points total and amount to be paid is printed on the receipt.

A program is to be written with the following specification:

- read the points total from the card
- process the amount spent
- output the amount to be paid and the new points total

A user-defined function `CalculatePoints` has already been coded to calculate the new points earned from the amount spent.

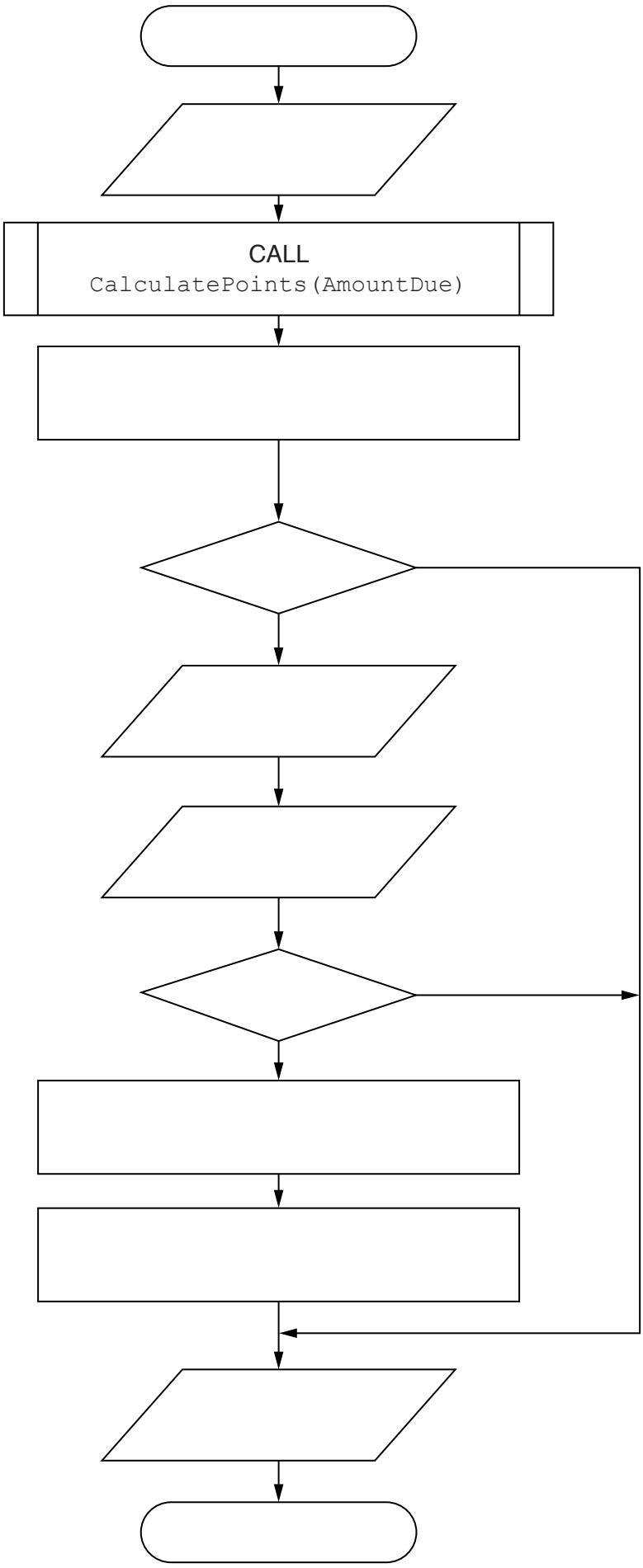
Study the following pseudocode:

```
INPUT AmountDue
NewPoints ← CalculatePoints(AmountDue)
PointsTotal ← PointsTotal + NewPoints

IF PointsTotal > 500
   THEN
      OUTPUT "Exchange points?"
      INPUT Response
      IF Response = "YES"
         THEN
            PointsTotal ← PointsTotal - 500
            AmountDue ← AmountDue - 1
      ENDIF
 ENDIF

 OUTPUT AmountDue, PointsTotal
```

The algorithm is also to be documented with a program flowchart.

Complete the flowchart by:

- filling in the flowchart boxes
- labelling, where appropriate, lines of the flowchart



[6]

4 The standard pack of playing cards has four suits – called Clubs, Diamonds, Hearts and Spades. Each card has a value shown by its number or a name: 1 (Ace), 2, 3, … 10, 11 (Jack), 12 (Queen), 13 (King). The pack of cards has one combination for each suit and value.

A program is to be written which simulates a magician dealing all 52 cards from the card pack.

The program generates pairs of random numbers:

- the first, in the range 1 to 4, to represent the suit
- the second, in the range 1 to 13, to represent the card value

(a) Explain why the generation of 52 (4 suits x 13 card values) pairs of random numbers will not simulate the dealing of the complete pack.

.....

.....

.....[2]

(b) A representation of dealing out the cards is shown below:

Suit number	Card value												
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1 (Clubs)	F	F	F	F	F	F	F	F	F	F	(T)	F	F
2 (Diamonds)	F	F	F	F	F	F	F	F	F	F	F	F	F
3 (Hearts)	F	F	(T)	F	F	F	F	F	F	F	F	F	F
4 (Spades)	F	F	F	F	F	F	F	F	F	F	F	F	F

The table shows two cards have been dealt so far; the 3 of Hearts and the Jack of Clubs.

When each card is dealt, the appropriate cell changes from F to T.

The program will output the suit and the card value in the order in which the cards are dealt.

Question 4(b) continues on page 8.

The program design in pseudocode is produced as follows:

```
01  DECLARE SuitNum      : INTEGER
02  DECLARE CardValue    : INTEGER
03  DECLARE DealCount    : INTEGER
04  DECLARE NewCard      : BOOLEAN
05  DECLARE CardPack .....................................................................................
06
07  CALL InitialiseCardPack
08  DealCount ← 0
09  WHILE DealCount <> 52
10    NewCard ← FALSE
11    WHILE NewCard = FALSE
12       SuitNum ← RANDOM(1,4)    // generates a random number
13       CardValue ← RANDOM(1,13) // in the range given
14       IF CardPack[SuitNum, CardValue] = FALSE
15          THEN
16             CardPack[SuitNum, CardValue] ← TRUE
17             NewCard ← TRUE
18             OUTPUT SuitNum, CardValue
19       ENDIF
20    ENDWHILE
21    DealCount ← DealCount + 1
22  ENDWHILE
23
24  // end of main program
25
26  PROCEDURE InitialiseCardPack
27    DECLARE i : INTEGER
28    DECLARE j : INTEGER
29    FOR i ← 1 TO 4
30       FOR j ← 1 TO 13
31          CardPack[i, j] ← FALSE
32       ENDFOR
33    ENDFOR
34  ENDPROCEDURE
```

Study the pseudocode and answer the questions below:

Give the line number for:

(i) A statement which marks the end of a count controlled loop.

.....[1]

(ii) The declaration of a local variable.

.....[1]

(iii) The initialisation of a variable used as a counter, but not to control a 'count controlled' loop.

.....[1]

(iv) A statement which uses a built-in function of the programming language.

.....[1]

(c) Give the number of procedures used by the pseudocode.

.....[1]

(d) Copy the condition which is used to control a 'pre-condition' loop.

.....[1]

(e) Explain the purpose of lines `14` – `19` in the design.

.....

.....

.....

.....

.....

.....[2]

(f) Complete the declaration of the global variable at line `05`.

```
05 DECLARE CardPack .....
```
[1]

(g) Line `18` in the design shows which new card is dealt each time.

When an Ace, Jack, Queen or King is dealt, the output displays the number for that card, not the name of the card.

Card value	Card name
1	Ace
11	Jack
12	Queen
13	King

A new requirement is to display the name of the card, where appropriate.

Write a CASE structure using variable `CardValue`.

Assign to a new variable `CardName` either:

- the card value (2, 3, 4, 5, 6, 7, 8, 9 or 10)
- or where appropriate, the card name Ace, Jack, Queen or King

.....

.....

.....

.....

.....

.....

.....

.....[4]

5 A program is to process a set of integers.

The integers are stored in an array, `Num`. The first `N` elements are to be processed.

The pseudocode for this program is shown below:

```
FOR i ← 1 TO (N - 1)
   j ← 1
   REPEAT
      IF Num[j] > Num[j + 1]
         THEN
            Temp ← Num[j]
            Num[j] ← Num[j + 1]
            Num[j + 1] ← Temp
      ENDIF
      j ← j + 1
   UNTIL j = (N - i + 1)
ENDFOR
```

(a) (i) Trace the execution of the pseudocode for the value `N` = 5 and the given array of integers.

N	i	j	Temp		Num 1	2	3	4	5
5					11	16	13	7	8

[8]

(ii) State the purpose of the algorithm.

.....

.....[1]

(iii) Describe what evidence from the trace table suggests that the given pseudocode is inefficient.

.....

.....[1]

(b) Complete the identifier table documenting the use of each of the variables.

Identifier	Data type	Description
Num	ARRAY[1:100] OF INTEGER	The array of numbers.
N		
i		
j		
Temp		

[5]

6 Some pseudocode statements follow which use the following built-in functions:

```
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("Barcelona", 3)` returns 'r'.

```
CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER
```
returns the number of characters in the string `ThisString`.
For example: `CHARACTERCOUNT("South Africa")` returns 12.

(a) Study the following pseudocode statements.

Give the values assigned to variables `x` and `y`.

(i) `x ← CHARACTERCOUNT("New Delhi") + 3` `x` ..... [1]

(ii) `y ← ONECHAR("Sri Lanka", 5)` `y` ..... [1]

(b) A program is to be written as follows:

- the user enters a string
- the program will form a new string with all <Space> characters removed
- the new string is output

```
NewString ← ""
INPUT InputString

j ← CHARACTERCOUNT(InputString)
FOR i ← 1 TO j
   NextChar ← ONECHAR(InputString, i)
   IF NextChar <> ""
      THEN
         // the & character joins together two strings
         NewString ← NewString & NextChar
   ENDIF
ENDFOR

OUTPUT NewString
```

(i) Complete the identifier table below.

Identifier	Data type	Description
`InputString`	`STRING`	The string value input by the user

[4]

(ii) An experienced programmer suggests this pseudocode would be best designed as a function.

Complete the re-design of the pseudocode as follows:

The main program:

- the user enters `MyString`
- the function is called and the changed string is assigned to variable `ChangedString`

The function:

- has identifier `RemoveSpaces`
- has a single parameter
- will include the declaration for any local variables used by the function

```
// main program
INPUT MyString
ChangedString←RemoveSpaces(.....)
OUTPUT ChangedString

// function definition

FUNCTION RemoveSpaces(.....) RETURNS .....

   .....

   .....

   .....

   .....

   j ← CHARACTERCOUNT(InputString)
   FOR i ← 1 TO j
      NextChar ← ONECHAR(InputString, i)
      IF NextChar <> " "
         THEN
            // the & character joins together two strings
            NewString ← NewString & NextChar
      ENDIF
   ENDFOR

   .....
ENDFUNCTION
```

[7]

7 ASCII character codes are used to represent a single character.

Part of the code table is shown below.

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

Some pseudocode statements follow which use these built-in functions:

```
CHARACTERCOUNT(ThisString : STRING) RETURNS INTEGER
```
returns the number of characters in the string `ThisString`.
For example: `CHARACTERCOUNT("South Africa")` returns 12.

```
CHR(ThisInteger : INTEGER) RETURNS CHAR
```
returns the character with ASCII value `ThisInteger`.
For example: `CHR(66)` returns 'B'.

```
ASC(ThisCharacter : CHAR) RETURNS INTEGER
```
returns the ASCII value for character `ThisCharacter`.
For example: `ASC('B')` returns 66.

(a) Give the values assigned to the variables `A`, `B`, `C` and `D`.

The & operator is used to concatenate two strings.

The expression could generate an error; if so, write ERROR.

```
Num1 ← 5
A ← ASC('F') + Num1 + ASC('Z')
B ← CHR(89) & CHR(69) & CHR(83)
C ← CHARACTERCOUNT(B & "PLEASE")
D ← ASC(ONECHAR("CURRY SAUCE", 7))
```

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

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

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

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

(b) A program is to be written to input a message string and then encrypt the message.

Study the following pseudocode:

```
OUTPUT "Enter message"
INPUT MyMessage
EncryptString ← ""

FOR i ← 1 TO CHARACTERCOUNT(MyMessage)
   NextNum ← ASC(ONECHAR(MyMessage, i) + 3)
   EncryptString ← EncryptString & CHR(NextNum)
ENDFOR

OUTPUT EncryptString
```

(i) Write the above pseudocode algorithm as **program code**.

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 .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[8]

(ii) Describe the encryption algorithm used to encrypt the message string entered by the user.

.....

.....

.....

.....[2]