Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER		CANDIDATE NUMBER	

COMPUTER SCIENCE **9608/22**

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.

[Turn over

There is an **Appendix** on pages 18 and 19. Some questions will refer you to this information.

1   A number of players take part in a competition. The competition consists of a number of games. Each game is between two players. The outcome of a game is that each player is awarded a grade (A, B, C or D). Each grade has an associated number of points as shown in the table below.

Grade	Points
A	0
B	1
C	3
D	5

The points total for all players is recorded. After each game is completed, the total number of points for each player is updated.

For example:

*   before the game between Ryan and Karina, Ryan's total is 5 points and Karina's total is 3 points
*   the result of the game between Ryan and Karina is: Ryan achieved grade B, Karina achieved grade D
*   the players' points totals are updated to: Ryan has 6 and Karina has 8

When a player's points total reaches 12 or higher, that player is removed from the competition.

A programmer will write a program to update the player total after a game.

The program will output:

*   the player's updated points total
*   the message 'ELIMINATED' if the player is removed from the competition.

The programmer designs the identifier table below:

Identifier	Data type	Description
PlayerName	STRING	Name of the player
PlayerGameGrade	CHAR	Game grade for the player
PointsTotal	INTEGER	Current player points
SavePlayerTotal	procedure	Procedure has parameters PlayerName and PointsTotal and saves the updated player total
ReadPlayerTotal	function	Function has a parameter PlayerName and returns the current total for that player

(a) Complete the following program flowchart by:

*   filling in the boxes, using pseudocode where appropriate
*   labelling the lines of the flowchart, where necessary.

[9]

```
                    ╭─────────────╮
                    │    Start    │
                    ╰──────┬──────╯
                           │
                           ▼
              ╱─────────────────────╲
             ╱                       ╱
            ╱                       ╱
           ╱───────────────────────╱
                       │
                       ▼
           ┌─┬───────────────────┬─┐
           │ │                   │ │
           └─┴───────────────────┴─┘
                       │
                       ▼
              ╱─────────────────────╲
             ╱                       ╱
            ╱                       ╱
           ╱───────────────────────╱
                       │
                       ▼
              ◇─────────────────◇
             ◇                   ◇────────┐
              ◇─────────────────◇         │      ┌──────────────────┐
                       │                  ├────▶ │                  │───┐
                       │                  │      └──────────────────┘   │
                       │                  │      ┌──────────────────┐   │
                       │                  ├────▶ │                  │──▶│
                       │                  │      └──────────────────┘   │
                       │                  │      ┌──────────────────┐   │
                       │                  └────▶ │                  │──▶│
                       │                         └──────────────────┘   │
                       ◀──────────────────────────────────────────────-┘
                       │
                       ▼
              ╱─────────────────────╲
             ╱                       ╱
            ╱                       ╱
           ╱───────────────────────╱
                       │
                       ▼
           ┌─┬───────────────────┬─┐
           │ │                   │ │
           └─┴───────────────────┴─┘
                       │
                       ▼
              ◇─────────────────◇  Yes
             ◇                   ◇────────────┐
              ◇─────────────────◇             │
                  │ No                        ▼
                  │               ╱─────────────────────╲
                  │              ╱                       ╱
                  │             ╱                       ╱
                  │            ╱───────────────────────╱
                  ◀───────────────────┘
                  │
                  ▼
           ╭─────────────╮
           │    Stop     │
           ╰─────────────╯
```
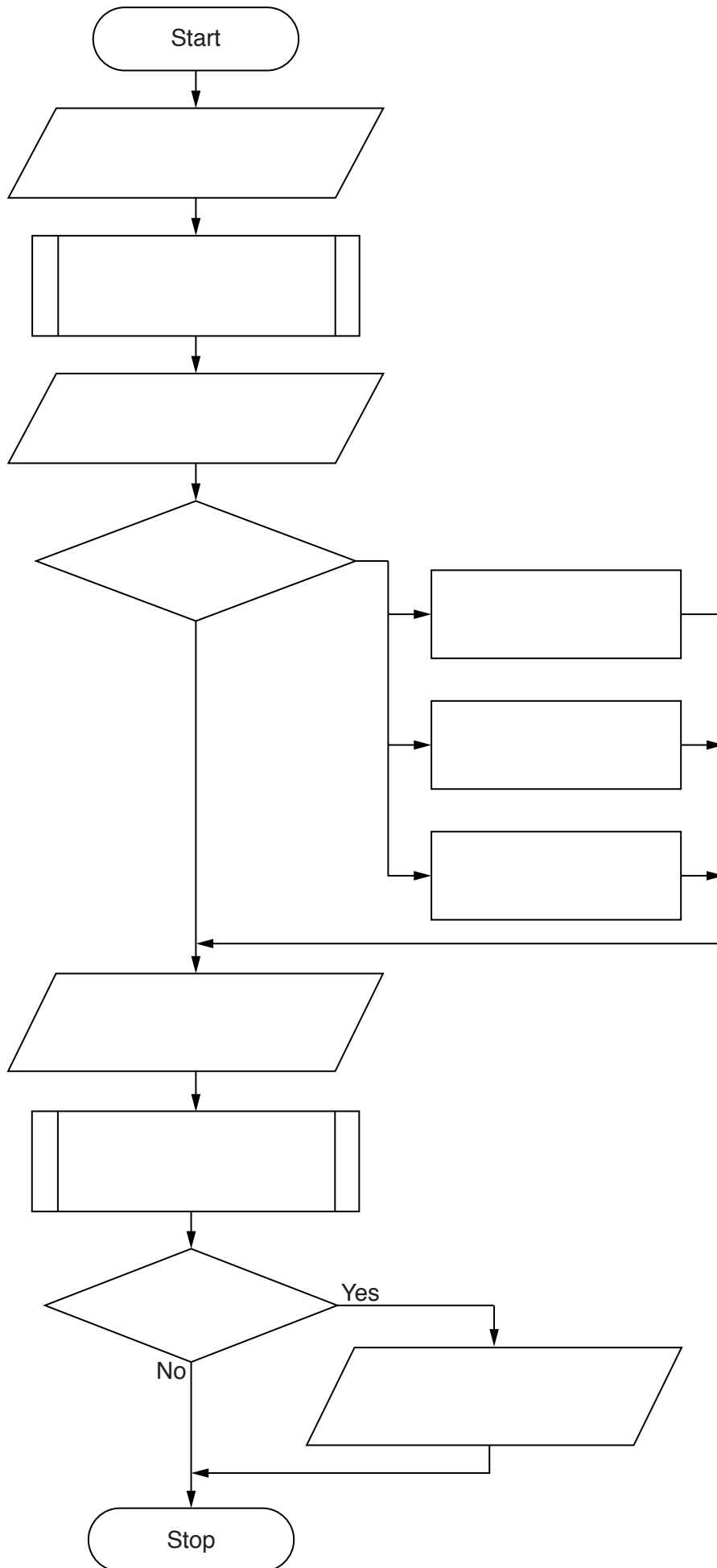
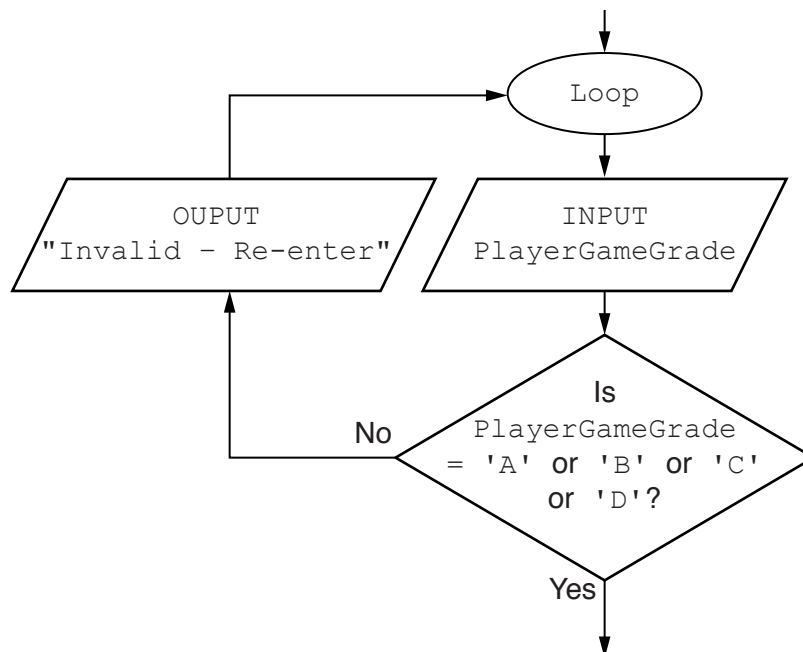**(b)** Test data is to be produced to test the flowchart.

Complete the table of test data below to show **five** tests that should be used to test different paths through the flowchart.

Test data		Expected results	
PointsTotal	PlayerGameGrade	Updated PointsTotal	Output

[5]

(c) The programmer amends the design to validate the value of player game grade that the user inputs.

The amended part of the flowchart is shown below.

Write the equivalent **pseudocode** using a pre-condition loop, for this part of the amended flowchart.

....

....

....

....

....

....

....

....

....

.... [3]

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

   (a) Give the value of the variables x, y and z for the following sequence of statements.

   (i) x ← ONECHAR("Barcelona", 5)   x .... [1]

   (ii) y ← ONECHAR("Cool", 1) & ONECHAR("Ball", 2) & "t-food"

                                                           y .... [1]

   (iii) Temp1 ← "13"
            Temp2 ← ONECHAR("One-2-One", 5)
            z ← TONUM(Temp2 & Temp1)   z .... [1]

A computer program is to simulate the reading and processing of a string of characters from an input device.

The character string consists of:

- a number of digit characters
- one or more <*> characters, each used as a separator
- a final <#> character.

A typical input character sequence, stored as InputString is:

                    13*156*9*86*1463*18*#

Study this pseudocode.

```
01 DECLARE Numbers ARRAY [1:100] OF INTEGER
02 DECLARE InputString      : STRING
03 DECLARE NextChar         : CHAR
04 DECLARE NextNumberString : STRING
05 DECLARE i                : INTEGER    // Numbers array index
06 DECLARE j                : INTEGER    // InputString index
07
08 OUTPUT "String ... "
09 INPUT InputString
10 j ← 1
11 NextChar ← ONECHAR(InputString, j)
12
13 i ← 1
14 WHILE NextChar <> '#'
15    NextNumberString = ""
16    WHILE NextChar <> '*'
17      NextNumberString ← NextNumberString & NextChar
18      j ← j + 1
19      NextChar ← ONECHAR(InputString, j)
20    ENDWHILE
21
22    // store the next integer to the array
23    Numbers[i] ← TONUM(NextNumberString)
24    i ← i + 1
25    j ← j + 1
26    NextChar ← ONECHAR(InputString, j)
27 ENDWHILE
28
29 CALL DisplayArray()
```

   (b) Write the line number for:

    (i) A statement which declares a global variable used to store a single character. .... [1]

    (ii) A statement which runs code written as a procedure.   .... [1]

    (iii) A statement which indicates the start of a 'pre-condition' loop.   .... [1]

    (iv) A statement which increments a variable.   .... [1]

   (c) Copy the condition which is used to control the inner loop.

    .... [1]

(d) (i) Complete the trace table below for the given pseudocode as far as line 27.

The input string is: `23*731*5*#`

i	j	NextChar	NextNumberString	Numbers		
				1	2	3
1	1	'2'				
			""			
			"2"			
	2	'3'	"23"			
	3	'*'		23		

[5]

(ii) Explain what this algorithm does.

....

....

....

.... [2]

3 Radhika mostly studied the high-level programming language XYZ at university.
She has been working in her first job for two years using language XYZ.
She applied for a new job which stated:

"The majority of the development work is done using language ABC."

(a) Radhika was interviewed for the job. Part of the interview process was to study some program code written in language ABC.

```
11  settype($TimesTable, Integer);
12  settype($upTo, Integer);
13  settype($Posn, Integer);
14  settype($Product, Integer);
15  $TimesTable = 7;
16  $UpTo = 10;
17
18  $Posn = 1
19  While ($Posn < $UpTo + 1)
20  {
21  $Product = $Posn * $TimesTable;
22  Echo $Posn . ' X' . $TimesTable . ' = ' . $Product . "<br>";
23  $Posn = $Posn + 1;
24  }
```

Answer the following questions taken from the interview.

(i) State what the `settype` keyword does in this language.

....

.... [1]

(ii) Name **one** variable that the code uses.

.... [1]

(iii) Give a line number for an assignment statement.

.... [1]

(iv) Line `19` is the start of a pre-condition loop.

State the syntax that language ABC uses to indicate which statements must be executed inside a loop.

.... [1]

(b) (i) Describe what is meant by a transferable skill.

....

....

.... [2]

(ii) Give **two** examples which suggest that programming in a high-level language is a transferrable skill.

1 ....

....

2 ....

.... [2]

4 A social club runs a weekly prize draw. A member can buy a ticket for $2. Each week, the club sells up to 150 tickets, with consecutive numbers starting from 1. There is a main cash prize of $120 and a further three prizes of $10. The computer program will generate the winning numbers.

Throughout this question, you will need to refer to the list of pseudocode functions in the **Appendix**.

(a) Write **pseudocode** to show how the RND() function can be used to generate a single integer in the range 1 to 150.

....

.... [3]

(b) Write **program code** to generate and output four winning numbers.
Ignore the issue that duplicate numbers may be generated.

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.

....

....

....

....

....

....

.... [4]

(c) The prize draw is a success. After six months, the club decided to sell an unlimited number of tickets each week. In any week, the total number of prizes may vary.

The programmer modifies the code written in **part(b)**.

The revised program will use a function to generate a single winning number. Therefore, in a week when the club offers six prizes, the program will use the function six times to generate the six winning numbers.

The function has identifier GenerateNumber and:

- has a single parameter for the number of tickets sold that week
- returns a single winning number.

Write **program code** for the function header.

Programming language ....

....

.... [3]

(d) The organisers do not want the same number to be drawn more than once in a week. To deal with the issue of duplicate numbers being generated, the program in **part (b)** will require adaptive maintenance.

(i) Describe what is meant by adaptive maintenance.

....

....

....

.... [2]

(ii) The club does not want the program to output the same number more than once in a week.

For each winning number, outline the extra steps that the program must have. Include any data structure that the program needs.
Do not write pseudocode or program code.

....

....

....

....

.... [3]

5 A team keeps a record of the scores made by each of their eight players in a number of games.

The data in the two tables below shows:

- the scores of the eight players after twenty games
- the eight player names.

	1	2	3		8
1	12	17	67		31
2	35	82	44		29
3	61	39	80		17
4	81	103	21		11
5	56	0	98		4
...					
19	45	6	81		77
20	12	11	3		6

1	Vorma
2	Ravi
3	Chada
4	Nigam
5	Bahri
6	Smith
7	Goyal
8	Lata

The team wants a computer program to input and record the player data.

(a) A programmer designs the following pseudocode for the input of a player's score from one game.

```
01  INPUT GameNumber
02  INPUT PlayerNumber
03  INPUT PlayerGameScore
04  PlayerScore[GameNumber, PlayerNumber] ← PlayerGameScore
```

Describe the data structure the programmer has used for the storage of all player scores.

.... [2]

(b) The player names are permanently stored in a text file NAMES.TXT, with one name on each line. The player names will be read by the program and stored in a 1D array.

The design given in **part (a)** will be expanded so that the user is prompted for the player name instead of the player number. Step 02 now becomes:

```
02.1   Read the player names from file NAMES.TXT into the array PlayerName
02.2   INPUT ThisPlayerName
02.3   Search the PlayerName array for ThisPlayerName to find the PlayerNumber
```

(i) State the computing term for the expansion of one or more steps in the original design.

.... [1]

(ii) Write the **program code** for step 02.1

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 ....

....

....

....

....

....

....

....

....

....

....

....

.... [4]

(iii) Program code is to be designed and written for step `02.3`

The program will use these identifiers:

Identifier	Data type	Description
PlayerName	ARRAY[1 : 8] OF STRING	Stores the player names (read from the file)
ThisPlayerName	STRING	Input by the user (step `02.2`)
Found	BOOLEAN	Flags when `ThisPlayerName` is found when searching the `PlayerName` array
i	INTEGER	Array index

Write **program code** to carry out the linear search for step `02.3`

There is no requirement to declare or comment on variables used.

Programming language ....

....

....

....

....

....

....

....

....

....

....

....

.... [4]

(c) The team wants the program to produce a report, with the following specification.

The program outputs the total number of player scores that are:

- 50 and over but less than 100
- 100 or higher.

You can assume that before the section runs, the program has assigned all eight player scores to the `PlayerScore` data structure.

A first attempt at the pseudocode is shown below:

```
01  Total50 ← 0
02  Total100 ← 0
03  FOR PlayerIndex  ← 1 TO 8
04    FOR GameIndex ← 1 TO 20
05      IF PlayerScore[GameIndex, PlayerIndex] > 100
06        THEN
07          Total100 ← Total100 + 1
08        ELSE
09          IF PlayerScore[GameIndex, PlayerIndex] > 50
10            THEN
11              Total50 ← Total50 + GameIndex
12          ENDIF
13        ENDIF
14    ENDFOR
15  ENDFOR
16  OUTPUT Total50
17  OUTPUT Total100
```

(i) Describe the control structure used in lines `03` and `04` and lines `14` and `15`.

....

....

.... [2]

(ii) Consider the following two statements.

Write either TRUE **or** FALSE next to each statement.

Statement	TRUE or FALSE
The pseudocode considers all the scores for a player, before progressing to the next player.	
The pseudocode considers all scores in a game, before progressing to the next game.	

[1]

(iii) The programmer has made logic errors in the design.

State a line number at which an error occurs.

Explain the error or write the corrected pseudocode statement.

Line number ....

Explanation ....

.... [1]

6 Study the sequence of pseudocode statements.

```
CONST  a = 3.2   : REAL
DECLARE x, y, z, Answer1, Answer2, Answer3  : REAL
DECLARE p, q     : BOOLEAN
x ← 3
x ← x + 7
y ← 6
Answer1 ← 2 * (a + y)
z ← 6
Answer2 ← y ^ 2 + 5
p ← TRUE
q ← NOT(NOT(p))
Answer3 ← y + a * 2
```

Give the final value assigned to each variable.

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

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

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

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

(v) `Answer3` .... [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

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

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

BLANK PAGE