Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME									
CENTRE NUMBER						CANDIDATE NUMBER			

COMPUTER SCIENCE **9608/02**

Paper 2 Fundamental Problem-solving and Programming Skills **For Examination from 2015**

SPECIMEN PAPER

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

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.

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

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.

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

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

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

Programming language used: .....

1 A program is to be written to enter and display the result of a cricket match. The winning team is the one scoring the most runs.

The structured English description of the problem is shown here. It assumes the scores are not equal.

```
INPUT HomeTeamName
INPUT HomeRuns
INPUT AwayTeamName
INPUT AwayRuns
SUBTRACT AwaysRuns FROM HomeRuns STORE AS RunDifference
CALCULATE the winning team STORE AS WinningTeamName
OUTPUT WinningTeamName and RunDifference
```

Typical output is shown.

```
Chargers
123
Tigers
136
Winning team was Tigers who scored 13 more runs
```

Write **program code** from this structured English.

[9]

2 A lottery game draws six numbers between 1 and 50, which are all different.

A computer program is to be developed to simulate the drawing of the six numbers.

A built-in function is available `RND()` which generates a random number between 0 and 0.99999999

The incomplete algorithm which follows takes no account of the numbers generated each time, so duplicates are possible.

(a) (i) Complete the entries in the table.

Identifier	Data Type	Explanation
`Counter`	.....	Loop counter
`NextNumber`	`INTEGER`	....

[2]

(ii) Complete the **pseudocode** using the identifiers.

```
FOR .....
    NextNumber ← .....
    OUTPUT NextNumber
    .....
OUTPUT "That completes the draw"
```

[3]

(b) Write **program code** for your completed algorithm for **part (a)(ii)**.

.....

.....

.....

.....

.....

.....

..... [3]

(c) The algorithm must be extended so that duplicate numbers are identified; that is, when a number is drawn which has already been drawn, the latest selection is ignored. The program continues until six different numbers are generated.

(i) Explain why a FOR loop structure is no longer suitable.

.....

..... [1]

(ii) Show, using **pseudocode**, the structure you will replace it with. Do **not** attempt to rewrite the algorithm in **part (a)(ii)**.

.....

..... [1]

The program designer decides on the method to identify duplicates. An array is used with upper bound 50. For example, the array cell with subscript 37 indicates whether or not the number 37 has been drawn.

The variables to be used are defined here:

Identifier	Data Type	Description
`NumberDrawn`	`ARRAY[50] : BOOLEAN`	FALSE – indicates the number has not yet been drawn TRUE – indicates the number has been drawn
`Index`	`INTEGER`	used as the subscript/index for the array

(iii) Write **pseudocode** to initialise the `NumberDrawn` array and code this as a procedure `InitialiseNumberDrawn`.

.....

.....

.....

.....

..... [3]

(iv) Complete the **pseudocode** to generate the six different numbers. The programmer has previously written a user-defined function `GenerateNumber` to generate a number between 1 and 50. This function is used in the algorithm.

Identifier	Data Type	Description
`Generated`	`INTEGER`	count of the number of different numbers drawn

```
CALL InitialiseNumberDrawn

Generated ← 0

.....// start of loop

   NextNumber ← GenerateNumber()

   IF NumberDrawn [.....] = .....

      THEN

         OUTPUT NextNumber

         Generated ← .....

         NumberDrawn [.....] ← .....

      ENDIF

.....// end of loop

OUTPUT "That completes the draw"
```

[6]

(v) Using the table, show the contents of the `NumberDrawn` array after the pseudocode in **part (iv)** is partially run and the loop has iterated five times, generating the number sequence 3, 47, 9, 47, 42.

	NumberDrawn
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
…	
39	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	
50	

[3]

(vi) Write the sequence of output from these five iterations.

.....

[1]

3 A football club is owned by its fans and currently has 3089 members. The data about members is held in a file `MEMBERS.DAT`. Each member is given a member number.

For each of the 23 home games, there is a prize draw with the member drawn gifted $100.

Once a member is a winner, they cannot win again.

The lucky member number is to be selected each week by a computer program. Hence, for the draw for the first week, it will generate a member number between 1 and 3089.

(a) The programmer will code a user-defined function `GenerateNumber` with this function header.

```
FUNCTION GenerateNumber (NoOfMembers : INTEGER) RETURNS INTEGER
         ──────1──────   ──────2──────   ──3──   ──────4──────
```

(i) Explain the various parts of the function header.

1. .....

.....

2. .....

.....

3. .....

.....

4. .....

..... [4]

(ii) This statement is used in the pseudocode.

```
PossibleWinner ← GenerateNumber(3089)
```

Explain this statement.

.....

..... [1]

The MEMBERS.DAT file is a text file with the following structure.

Each member's data is a single line of text, consisting of:

- four characters for the member number (fixed length)
- a <Space> character
- member name (variable length).

MEMBERS.DAT

```
0001 LANGO AMARA
0002 MUHAMMED JAMILA
0003 KURD BILAL
      ⌒
1064 MUHAMMED ABDUL
1065 BUTT WASIM
      ⌒
3089 DAR ZAFAR
```

A second serial file PREVIOUSWINNERS.DAT stores the member numbers of all past winners. The diagram shows the file after five home games.

PREVIOUSWINNERS.DAT

```
3011
0089
1067
0865
0719
```

(b) Explain why these winners' data must be stored in a file.

.....

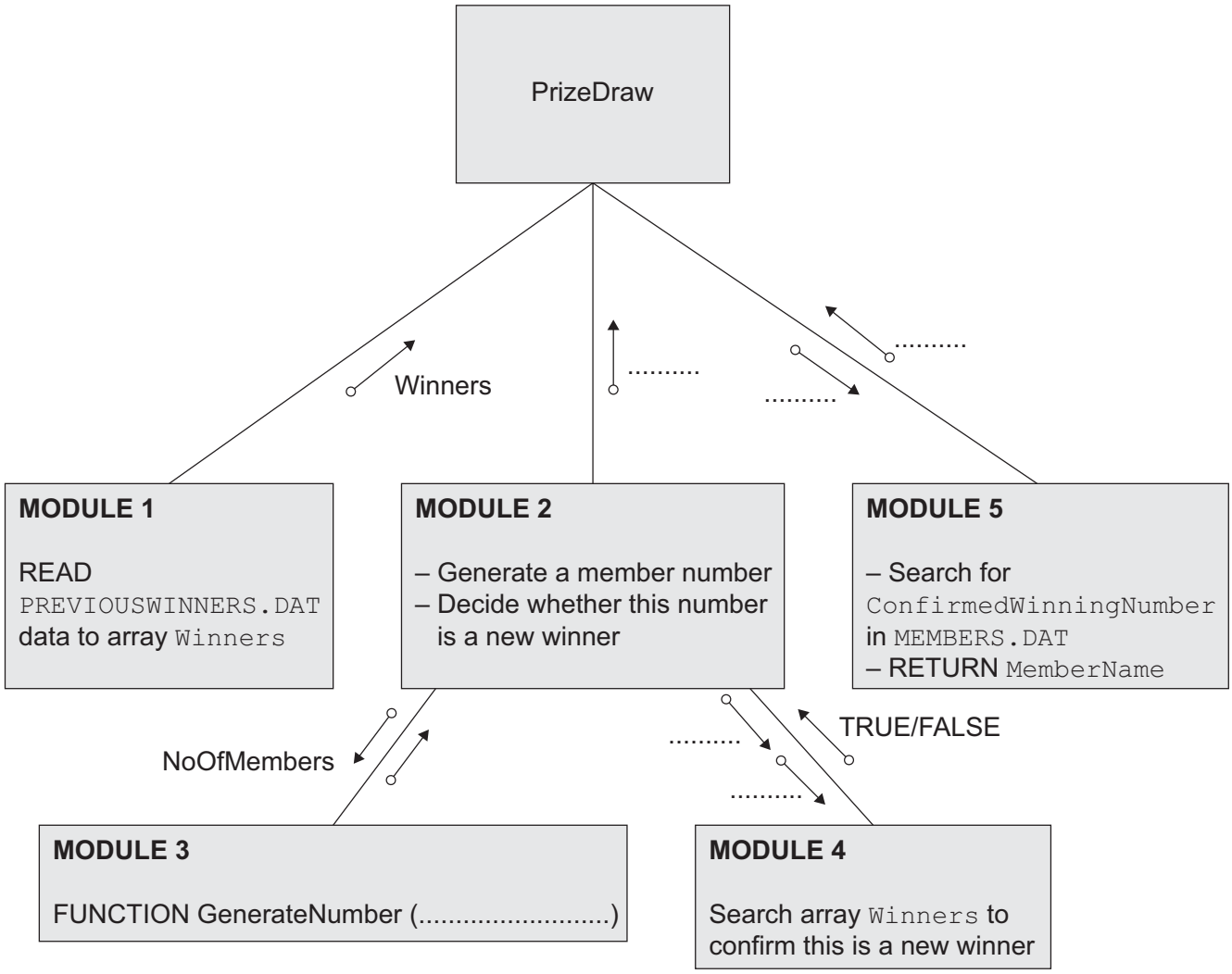
..... [2]

(c) An initial program specification is shown by the structure chart.

The programmer uses the identifier table shown here.

Identifier	Data Type	Description
`PREVIOUSWINNERS.DAT`	`TEXT FILE`	Text file storing the member numbers of all previous winners
`Winners`	`ARRAY[4000] : STRING`	Array to store the four-character member number
`PossibleWinner`	`INTEGER`	The new number generated for this draw (the program will check it has not been a previous winner)
`ConfirmedWinningNumber`	`INTEGER`	After the check, we can confirm it has not been drawn already. It has the same value as `PossibleWinner`

Complete the structure chart. There are **six** missing pieces of information.



[6]

(d) For MODULE 1, the programmer writes pseudocode as follows:

```
PROCEDURE ReadPreviousWinnersFile
   OPENFILE PREVIOUSWINNERS.DAT FOR INPUT
   REPEAT
      READ line of text and store at next position in array Winners
   UNTIL end of the file
   OUTPUT "File contents now read to array"
ENDPROCEDURE
```

(i) Show any other variable(s) which you need to add to the table given in **part (c)** to code the procedure.

Identifier	Data Type	Description

[3]

(ii) Write **program code** to implement this procedure `ReadPreviousWinnersFile`.

[8]

(e) As each member's data is a line of text, it will be stored by the program. The program uses it as a string.

Assume the programming language has built-in functions defined as follows:

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

Returns the given number of characters from `ThisString` starting with the first character.

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

Returns the given number of characters from `ThisString` starting counting from the final character and moving left.

```
MID(ThisString: STRING, Number1: INTEGER, Number2: INTEGER)
                                                RETURNS STRING
```

Returns a sub-string of `ThisString` as follows: starts at position `Number1` and then retains `Number2` characters.

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

Returns the number of characters in `ThisString`.

Assume the first line from the `MEMBERS.DAT` file was stored by a program as:

```
MemberData ← "0001 LANGO AMARA"
```

Using the functions given, write statements to do the following:

(i) Calculate the length of the string.

`DataLength` ← ..... [1]

(ii) Extract the member number.

`MemberNumber` ← ..... [1]

(iii) Extract the member name.

`MemberName` ← ..... [1]

4 A student writes some code which uses ASCII character codes.

Character	Denary	Character	Denary	Character	Denary
		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

Assume the programming language has these built-in functions available.

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

`ThisNumber` is the denary ASCII code for the character returned by the function.

E.g. `CHR(65)` returns 'A'.

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

The function returns the denary ASCII value of the character `ThisCharacter`.

E.g. `ASC('A')` returns 65.

(a) What is the value of these expressions?

(i) `CHR(80)`

..... [1]

(ii) `ASC('J') + 13`

..... [1]

(b) What is the value of the variable `Answer`, following the execution of these two statements?

```
ThisValue ← 69 - ASC('B')
Answer ← ASC('W') - ThisValue
```

Answer = ... [1]

(c) The student is interested in how simple encryption could be applied to a text message.

One of the simplest forms of encryption is a method of 'substitution' where each character has a unique substitute character.

The student uses this method with the following character substitutions:

Message character	A	B	C	D	E	F	G	H	I	J	K	L	M
Substitute character	P	L	F	N	O	C	Q	U	D	Z	V	G	I

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
X	M	W	J	B	K	E	A	H	S	Y	R	T

Assume all messages are made up from the upper-case characters only.

Show the string after the message `ATSEVEN` is encrypted.

..... [1]

(d) The program inputs a message string from the keyboard, stored in the variable `MessageString`.

Identifier	Data Type	Description
`MessageString`	`STRING`	message string input by the user
`LengthMessageString`	`INTEGER`	the number of characters in `MessageString`
`Alphabet`	`ARRAY[26]: CHAR`	the alphabet of upper case characters in order
`Substitute`	`ARRAY[26]: CHAR`	the substitute character for each character in the `Alphabet` array.

(i) The programming language allows reference to individual characters, starting with index 1 as the first character.

E.g. `MessageString ← "YESICAN"`

`MessageString[3] = 'S'`

Write the algorithm in **pseudocode** to input a message string and output the encrypted string. Your algorithm will search the `Alphabet` array for each of the characters in `MessageString`.

You may need to use additional variables to those already given in the identifier table.

[10]

(ii) Describe a more efficient algorithm to the one written in **part (i)**.

.....

.....

.....

..... [2]