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 AwayRuns 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.
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.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Data Type</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter</td>
<td></td>
<td>Loop counter</td>
</tr>
<tr>
<td>NextNumber</td>
<td>INTEGER</td>
<td></td>
</tr>
</tbody>
</table>

(ii) Complete the pseudocode using the identifiers.

```
FOR
    ____________________________________________
    NextNumber ← ________________________________
    OUTPUT NextNumber
    ________________________________
    OUTPUT “That completes the draw”
```

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

```
__________________________________________________________
__________________________________________________________
__________________________________________________________
__________________________________________________________
__________________________________________________________
__________________________________________________________
```

[2]
(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.

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

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:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberDrawn</td>
<td>ARRAY[50] : BOOLEAN</td>
<td>FALSE – indicates the number has not yet been drawn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE – indicates the number has been drawn</td>
</tr>
<tr>
<td>Index</td>
<td>INTEGER</td>
<td>used as the subscript/index for the array</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generated</td>
<td>INTEGER</td>
<td>count of the number of different numbers drawn</td>
</tr>
</tbody>
</table>

CALL `InitialiseNumberDrawn`

 Generated ← 0

--------------------------------------------------------------- // start of loop

   NextNumber ← GenerateNumber()

   IF NumberDrawn [ ] = 0 THEN

      OUTPUT NextNumber

      Generated ← 0

   ENDIF

   NumberDrawn [ ] ← NextNumber

--------------------------------------------------------------- // end of loop

OUTPUT “That completes the draw”
(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.

<table>
<thead>
<tr>
<th>NumberDrawn</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>39</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>41</td>
</tr>
<tr>
<td>42</td>
</tr>
<tr>
<td>43</td>
</tr>
<tr>
<td>44</td>
</tr>
<tr>
<td>45</td>
</tr>
<tr>
<td>46</td>
</tr>
<tr>
<td>47</td>
</tr>
<tr>
<td>48</td>
</tr>
<tr>
<td>49</td>
</tr>
<tr>
<td>50</td>
</tr>
</tbody>
</table>

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

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

1. [4]

(ii) This statement is used in the pseudocode.

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

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

The programmer uses the identifier table shown here.

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

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

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

\[ \text{MemberData} \leftarrow \text{“0001 LANGO AMARA”} \]

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

(i) Calculate the length of the string.
\[
\text{DataLength} \leftarrow \text{LEN(MemberData)} \]

(ii) Extract the member number.
\[
\text{MemberNumber} \leftarrow \text{LEFT(MemberData, 4)} \]

(iii) Extract the member name.
\[
\text{MemberName} \leftarrow \text{MID(MemberData, 5, 10)} \]
A student writes some code which uses ASCII character codes.

<table>
<thead>
<tr>
<th>Character</th>
<th>Denary</th>
<th>Character</th>
<th>Denary</th>
<th>Character</th>
<th>Denary</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>73</td>
<td>R</td>
<td>82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>65</td>
<td>J</td>
<td>74</td>
<td>S</td>
<td>83</td>
</tr>
<tr>
<td>B</td>
<td>66</td>
<td>K</td>
<td>75</td>
<td>T</td>
<td>84</td>
</tr>
<tr>
<td>C</td>
<td>67</td>
<td>L</td>
<td>76</td>
<td>U</td>
<td>85</td>
</tr>
<tr>
<td>D</td>
<td>68</td>
<td>M</td>
<td>77</td>
<td>V</td>
<td>86</td>
</tr>
<tr>
<td>E</td>
<td>69</td>
<td>N</td>
<td>78</td>
<td>W</td>
<td>87</td>
</tr>
<tr>
<td>F</td>
<td>70</td>
<td>O</td>
<td>79</td>
<td>X</td>
<td>88</td>
</tr>
<tr>
<td>G</td>
<td>71</td>
<td>P</td>
<td>80</td>
<td>Y</td>
<td>89</td>
</tr>
<tr>
<td>H</td>
<td>72</td>
<td>Q</td>
<td>81</td>
<td>Z</td>
<td>90</td>
</tr>
</tbody>
</table>

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

\[
\text{CHR(ThisNumber: INTEGER) RETURNS CHAR} \\
\text{ThisNumber is the denary ASCII code for the character returned by the function.} \\
\text{E.g. CHR(65) returns ‘A’.
}
\]

\[
\text{ASC(ThisCharacter: CHAR) RETURNS INTEGER} \\
\text{The function returns the denary ASCII value of the character ThisCharacter.} \\
\text{E.g. ASC(‘A’) returns 65.
}
\]

(a) What is the value of these expressions?

(i) \( \text{CHR(80)} \)

.............................................................................................................................................. [1]

(ii) \( \text{ASC(‘J’) + 13} \)

.............................................................................................................................................. [1]

(b) What is the value of the variable \( \text{Answer} \), following the execution of these two statements?

\[
\text{ThisValue ← 69 – ASC(‘B’)} \\
\text{Answer ← ASC(‘W’) – ThisValue} \\
\text{Answer =} \\
\]

.............................................................................................................................................. [1]
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:

<table>
<thead>
<tr>
<th>Message character</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Substitute character</td>
<td>P</td>
<td>L</td>
<td>F</td>
<td>N</td>
<td>O</td>
<td>C</td>
<td>Q</td>
<td>U</td>
<td>D</td>
<td>Z</td>
<td>V</td>
<td>G</td>
<td>I</td>
</tr>
</tbody>
</table>

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

Show the string after the message ATSEVEN is encrypted.

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

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageString</td>
<td>STRING</td>
<td>message string input by the user</td>
</tr>
<tr>
<td>LengthMessageString</td>
<td>INTEGER</td>
<td>the number of characters in MessageString</td>
</tr>
<tr>
<td>Alphabet</td>
<td>ARRAY[26]: CHAR</td>
<td>the alphabet of upper case characters in order</td>
</tr>
<tr>
<td>Substitute</td>
<td>ARRAY[26]: CHAR</td>
<td>the substitute character for each character in the Alphabet array.</td>
</tr>
</tbody>
</table>
The programming language allows reference to individual characters, starting with index 1 as the first character.

E.g. \( \text{MessageString} \leftarrow \text{“YESICAN”} \)

\[ \text{MessageString}[3] = ‘S’ \]

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).