1 The files **WORDS1.TXT** and **WORDS2.TXT** store a list of single word computing terms used in a textbook.

In **WORDS1.TXT**, each entry has the following format:

<computing term>,<number>

An example of an entry (in **WORDS1.TXT**) is:

compiler,12

This means that after a complete scan of the textbook the word **'compiler'** was found **12** times.

Task 1.1

Write program code to find and output the term with the highest number of occurrences. Use the file **WORDS1.TXT** to test your program.

EVIDENCE 1: Your program code. **[8]**

EVIDENCE 2: Screenshot of output. **[1]**

In **WORDS2.TXT**, each entry has the following format:

<computing term>
<number>

An example of an entry (in **WORDS2.TXT**) is:

procedure
22

Task 1.2

Amend your program code so that if more than one term exists with the highest number of occurrences, all terms are reported. Use the file **WORDS2.TXT** to test your program.

EVIDENCE 3: Your program code. **[5]**

EVIDENCE 4: Screenshot of output. **[1]**

[Total: 15 marks]

2 A prime number is any whole number (integer) which has only two factors, 1 and itself. The exception to this rule is 1, which is not a prime number. The following is a pseudocode algorithm to determine if a positive integer N is prime.
The algorithm is both poorly designed and contains errors.

```
FOR x = 1 TO N
    IF (N mod x) IS EQUAL TO 0 THEN
        flag = False
    ELSE
        flag = True
    ENDIF
ENDFOR
```

Task 2.1

Write program code for this algorithm including all the improvements you would make to:

- correct the errors
- follow good programming practice
- make the algorithm more efficient

Test your program with **four** suitable positive integers, including 1 and 2.

EVIDENCE 5: Your program code. **[9]**

EVIDENCE 6: Using **four** suitable test data, show screenshots from running the program. **[4]**

Task 2.2

Re-design the program code to have a function **`IsPrime`** using the following specification:

```
FUNCTION IsPrime(number: INTEGER) RETURN BOOLEAN
```

This function should have a parameter which allows it to be used for any positive integer. Write also program code to make use of the `IsPrime` function to **generate the first 20 prime numbers**.

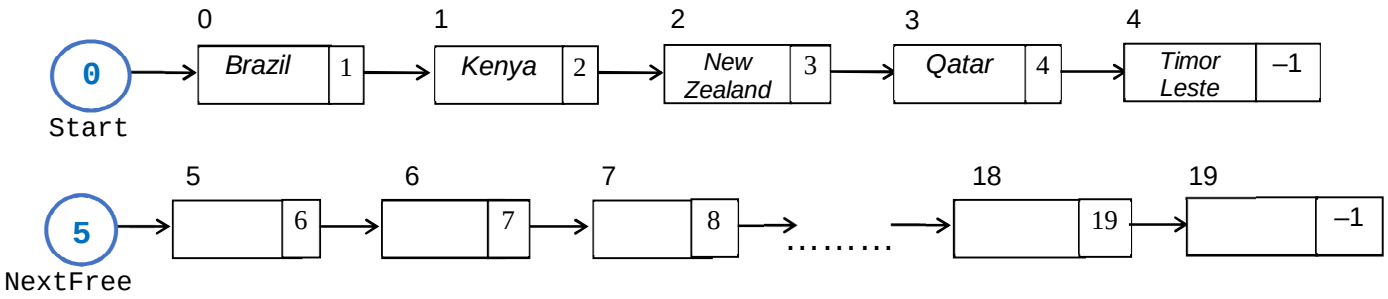
EVIDENCE 7: Your `IsPrime` function and program code to generate the first 20 prime numbers. **[5]**

EVIDENCE 8: Screenshot showing the output from running the program. **[2]**

[Total: 20 marks]

3 A program is to be written to store names of countries in alphabetical order using a linked list.

The diagram shows the linked list with 5 countries added and the unused nodes linked together.



The program will use nodes implemented as instances of the class **`ListNode`**. The class **`ListNode`** has the following properties:

Class: `ListNode`		
Properties		
Identifier	**Data Type**	**Description**
`Name`	`STRING`	The node's value for a country name
`Pointer`	`INTEGER`	The pointer for the node

A linked list is implemented as an instance of the class `LinkedList`. The class `LinkedList` has the following properties and methods:

Class: `LinkedList`		
Properties		
Identifier	**Data Type**	**Description**
`Node`	`ARRAY[20] of ListNode`	The linked list data structure – data value (Name) and pointers. Array index starts at 0. Null value is –1. Dataset has a maximum of 20 nodes.
`Start`	`INTEGER`	Index position of the node at the start of the linked list
`NextFree`	`INTEGER`	Index position of the next unused node
Methods		
`Initialise`	`PROCEDURE`	Sets all node data value (Name) to empty string. Set pointers to indicate all nodes are unused and linked. Initialise values for `Start` and `NextFree`.
`Insert`	`PROCEDURE`	Insert a new value into the linked list.
`Display`	`PROCEDURE`	Display the current state of array content and pointers in table form.
`IsEmpty`	`FUNCTION RETURNS BOOLEAN`	Test for empty linked list.
`IsFull`	`FUNCTION RETURNS BOOLEAN`	Test for no unused nodes.

Task 3.1

Write program code for the classes **`ListNode`** and **`LinkedList`**, including the **`Initialise`**, **`Display`**, **`IsEmpty`** and **`IsFull`** method.
The code should follow the specification given.
Do not write the **`Insert`** procedure yet.

Evidence 9: Your program code for the **`ListNode`** and **`LinkedList`** classes. **[12]**

Task 3.2

Write program code to create a **`LinkedList`** object and run **`Display`** procedure to show the content of the object, which includes the array of `ListNode`, `Start` and `NextFree`.

Evidence 10: Your **program code** & **screenshot** after creating and displaying `LinkedList` object. **[3]**

Task 3.3

Write program code to implement the **Insert** method for the **LinkedList** class that will insert a new value into the linked list in alphabetical order. The pseudocode is given below.

Evidence 11: Program code for **Insert** procedure. **[12]**

```
PROCEDURE Insert(new_value)
    IF nextfree = NULL THEN
        put("No space to insert") and EXIT //Check if free node exist
    nextfree.value ← new_value              //Store new_value in next free node
    IF start = NULL THEN                    //Insert into empty list
        holdfree ← nextfree.ptr
        nextfree.ptr ← NULL
        start ← nextfree
        nextfree ← holdfree
    ELSE
        IF new_value < start.value THEN     //Insert as first node of list
            holdfree ← nextfree.ptr
            nextfree.ptr ← start
            start ← nextfree
            nextfree ← holdfree
        ELSE
            previous ← start
            current ← start
            WHILE new_value > current.value AND current.ptr ≠ NULL
                                            //Search position to insert node
                previous ← current
                current ← current.ptr
            ENDWHILE
            IF new_value > current.value AND current.ptr = NULL THEN
                                            //Insert as last node of list
                holdfree ← nextfree.ptr
                current.ptr ← nextfree
                nextfree.ptr ← NULL
                nextfree ← holdfree
            ELSE                            //Insert in-between nodes
                holdfree ← nextfree.ptr
                previous.ptr ← nextfree
                nextfree.ptr ← current
                nextfree ← holdfree
            ENDIF
        ENDIF
    ENDIF
ENDPROCEDURE
```

Task 3.4

Write program code to use the **`Insert`** procedure by reading in the text from the file **`COUNTRIES.TXT`**.

Evidence 12: Program code that uses the **`Insert`** procedure. **[3]**

Evidence 13: Screenshot of state of array content and pointers by running **`Display`** procedure after insertion of text from file. **[1]**

Task 3.5

Write a method **`Query`** in the **`LinkedList`** class that:

- takes a country input by user,
- check if the country exists in the linked list,
- output appropriate message.

Evidence 14: Program code for **`Query`** method. **[7]**

Evidence 15: Screenshots of 2 test cases from running the **`Query`** method. **[2]**

[Total: 40 marks]

~ END OF PAPER ~