Open the document **evidence.doc**.

Make sure that your name, centre number and candidate number will appear on every page of this document. This document will contain your answers to each question.

Save this evidence document in your work area as:

evidence_ followed by your centre number_candidate number, for example: evidence_zz999_9999

A class declaration can be used to declare a record.

A list is an alternative to an array.

A source file is used to answer question 3. The file is called **`TreasureChestData.txt`**

1 An unordered linked list uses a 1D array to store the data.

Each item in the linked list is of a record type, `node`, with a field `data` and a field `nextNode`.

The current contents of the linked list are:

startPointer	0
emptyList	5

Index	**data**	**nextNode**
0	1	1
1	5	4
2	6	7
3	7	-1
4	2	2
5	0	6
6	0	8
7	56	3
8	0	9
9	0	-1

(a) The following is pseudocode for the record type `node`.

```
TYPE node
    DECLARE data : INTEGER
    DECLARE nextNode : INTEGER
ENDTYPE
```

Write program code to declare the record type `node`.

Save your program as **question1**.

Copy and paste the program code into **part 1(a)** in the evidence document.

[2]

(b) Write program code for the main program.

Declare a 1D array of type `node` with the identifier `linkedList`, and initialise it with the data shown in the table on page 2. Declare the pointers.

Save your program. Copy and paste the program code into **part 1(b)** in the evidence document.

[4]

(c) The procedure `outputNodes()` takes the array and `startPointer` as parameters. The procedure outputs the data from the linked list by following the `nextNode` values.

(i) Write program code for the procedure `outputNodes()`.

Save your program. Copy and paste the program code into **part 1(c)(i)** in the evidence document.

[6]

(ii) Edit the main program to call the procedure `outputNodes()`.

Take a screenshot to show the output of the procedure `outputNodes()`.

Save your program. Copy and paste the screenshot into **part 1(c)(ii)** in the evidence document.

[1]

(d) The function, `addNode()`, takes the linked list and pointers as parameters, then takes as input the data to be added to the end of the `linkedList`.

The function adds the node in the next available space, updates the pointers and returns `True`. If there are no empty nodes, it returns `False`.

(i) Write program code for the function `addNode()`.

Save your program. Copy and paste the program code into **part 1(d)(i)** in the evidence document.

[7]

(ii) Edit the main program to:

- call `addNode()`
- output an appropriate message depending on the result returned from `addNode()`
- call `outputNodes()` twice; once before calling `addNode()` and once after calling `addNode()`.

Save your program. Copy and paste the program code into **part 1(d)(ii)** in the evidence document.

[3]

(iii) Test your program by inputting the data value `5` and take a screenshot to show the output.

Save your program. Copy and paste the screenshot into **part 1(d)(iii)** in the evidence document.

[1]

(e) (i)The function `deleteNode()` which takes the linked list and pointers as parameters and let user input a data and delete the node with the data return True , if the data is not in the list return False.

Save your program, copy and paster the program code input part 1 (e)(i)

(ii) edit the main program, call deleteNode() two times, input data which can be deleted successfully and the data which is not in the list. and according the return value ,print out proper message .

Save your program, run the the program code and screenshot your main program and the testing result together and copy and past the screenshot input part 1 (e)(ii)