1 A program stores positive integers in a circular queue.

The queue is stored as a global 1D array of 20 integers with the identifier `Queue`. Each index is initialised with the data `-1`

The global variable `HeadPointer`, initialised to `-1`, points to the first element in the queue.
The global variable `TailPointer`, initialised to `-1`, points to the last element in the queue.
The global variable `NumberItems`, initialised to `0`, stores the number of items in the queue.

(a) Write program code to declare and initialise `Queue`, `HeadPointer`, `TailPointer` and `NumberItems`

Save your program as **Question1_J25**.

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

[2]

(b) The function `Enqueue()`:

- takes an integer as a parameter
- checks if the queue is full
- returns `FALSE` if the queue is full
- stores the parameter in the next position in the queue and returns `TRUE` if the queue is **not** full
- updates the appropriate pointers and `NumberItems`

Write program code for `Enqueue()`

Save your program.

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

[6]

(c) The main program:

- attempts to store each of the integers 1 to 25 (inclusive) in the queue in ascending numerical order using `Enqueue()`
- outputs the integer that was passed to `Enqueue()` and `"Successful"` if it was stored in the queue, or `"Unsuccessful"` if it was **not** stored in the queue.
  For example:
  - if the integer 5 is passed to `Enqueue()` and is stored in the queue, the output will be: `"5 Successful"`
  - if the integer 23 is passed to `Enqueue()` and is **not** stored in the queue, the output will be: `"23 Unsuccessful"`

Write program code for the main program.

Save your program.

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

[3]

(d) The function `Dequeue()` returns `-1` if the queue is empty. If the queue is **not** empty, the function returns the next item in the queue, updates the appropriate pointers and updates `NumberItems`

Write program code for `Dequeue()`

Save your program.

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

[6]

(e) **(i)** Write program code to extend the main program to call `Dequeue()` twice and output the return value each time.

Save your program.

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

[2]

(ii) Test your program.

Take a screenshot of the output(s).

Save your program.

Copy and paste the screenshot into part **1(e)(ii)** in the evidence document.

[1]