Name: Amirtha Ganesh Pugazhendhi

Structure & function of the Program:

1) There were 4 files required for running the assignment program namely
   1. **"LLQueue.c"** – This file consist of c programming codes for
      1. **Data Structre definition:** Node, Linked list, Head, Tail
      2. **Function definition:** Creating LLQueue , Inserting LLQueue, Deleting LLQueue, Printing the Llqueue, Counting the elements of LLQueue, Searching within LLQueue, Finding Maximum value of LLQueue, Finding Minimum value of LLQueue and Deleting & Freeing all LLQueue memory

   2. **"LLQueue.h"-** This file consist of c programming codes for **Typedef** and **Function declaration** namely,

      1) LLQueue *LLQ_create(); **2)** int LLQ_insert(LLQueue *LLQ, double data); **3)** double LLQ_delete(LLQueue *LLQ);

      4) double *LLQ_Search(LLQueue *LLQ, double data); **5)** double LLQ_minimum(LLQueue *LLQ);
      6) double LLQ_maximum(LLQueue *LLQ); **7)** unsigned int LLQ_count(LLQueue *LLQ); **8)** void LLQ_print(LLQueue *LLQ);

      9) void LLQ_free(LLQueue *LLQ);

   3. **"UnitTest.c"** – This file consist c programming codes running unit test:
      1. Creating the queue**. 2)** Inserting an item into an empty queue. **3)** Returning the maximum and minimum of a queue containing at least 4 elements.**4)** Searching for an item present at the beginning, middle, and end of the linked list for a queue containing at least 4 elements **5)** Searching for an item that is not present in a queue containing at least 4 elements (error case). **6)** Deleting all of the items in a queue containing at least 4 elements, one after the other, finally deleting the queue. **7)** Deleting an item from an empty queue (error case).

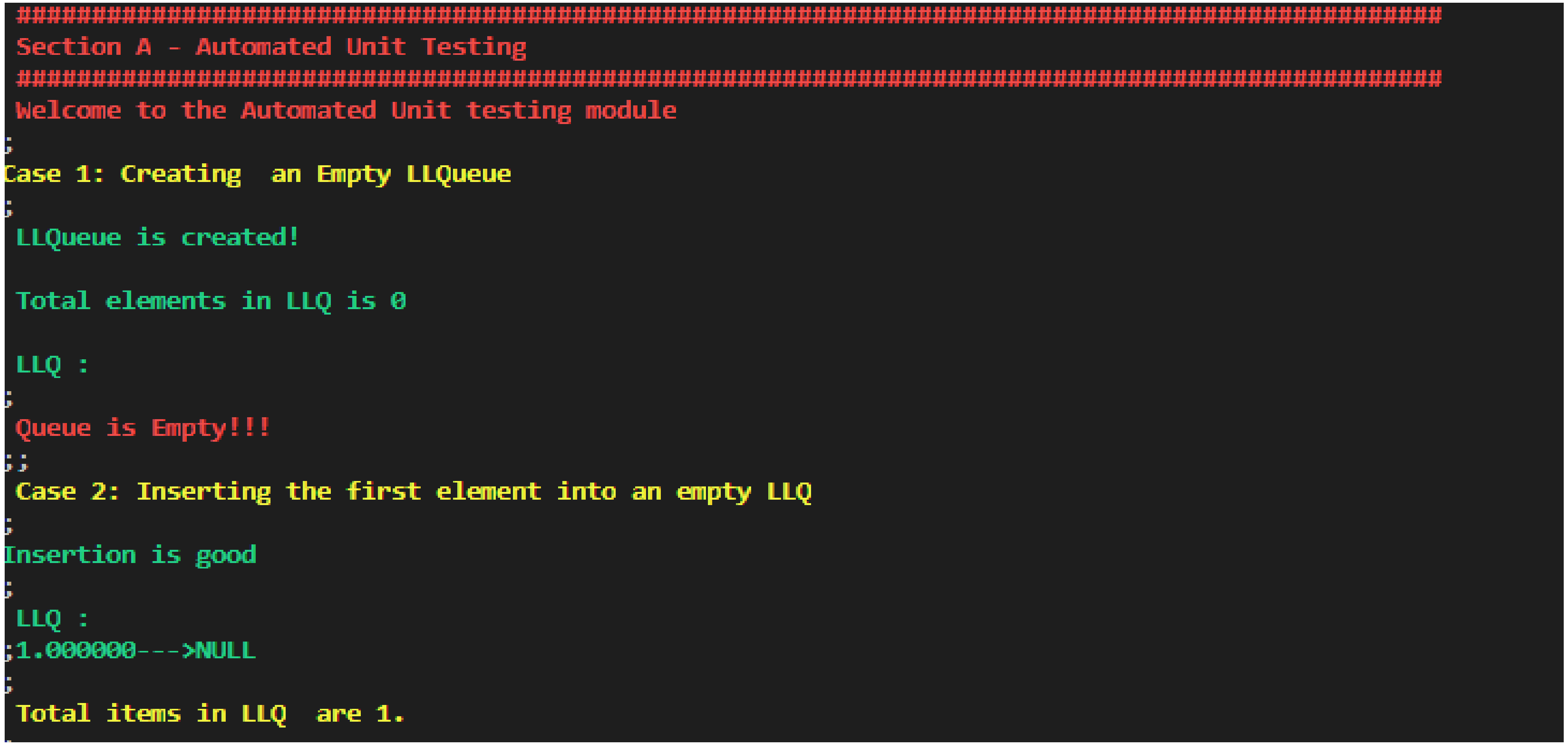
   4. **"makefile"-** This file consist of "makefile" programming codes to produce object files and executable for the assignment

Name: Amirtha Ganesh Pugazhendhi

Unit Testing was divided into **2 modules**

1) Automated Unit Testing module: This module is indented for the execution of unit test, defined in programming assignment question with **pre-determined test conditions** and **cases**. The most important and Critical output was creation of **" Empty LLQueue"**

output

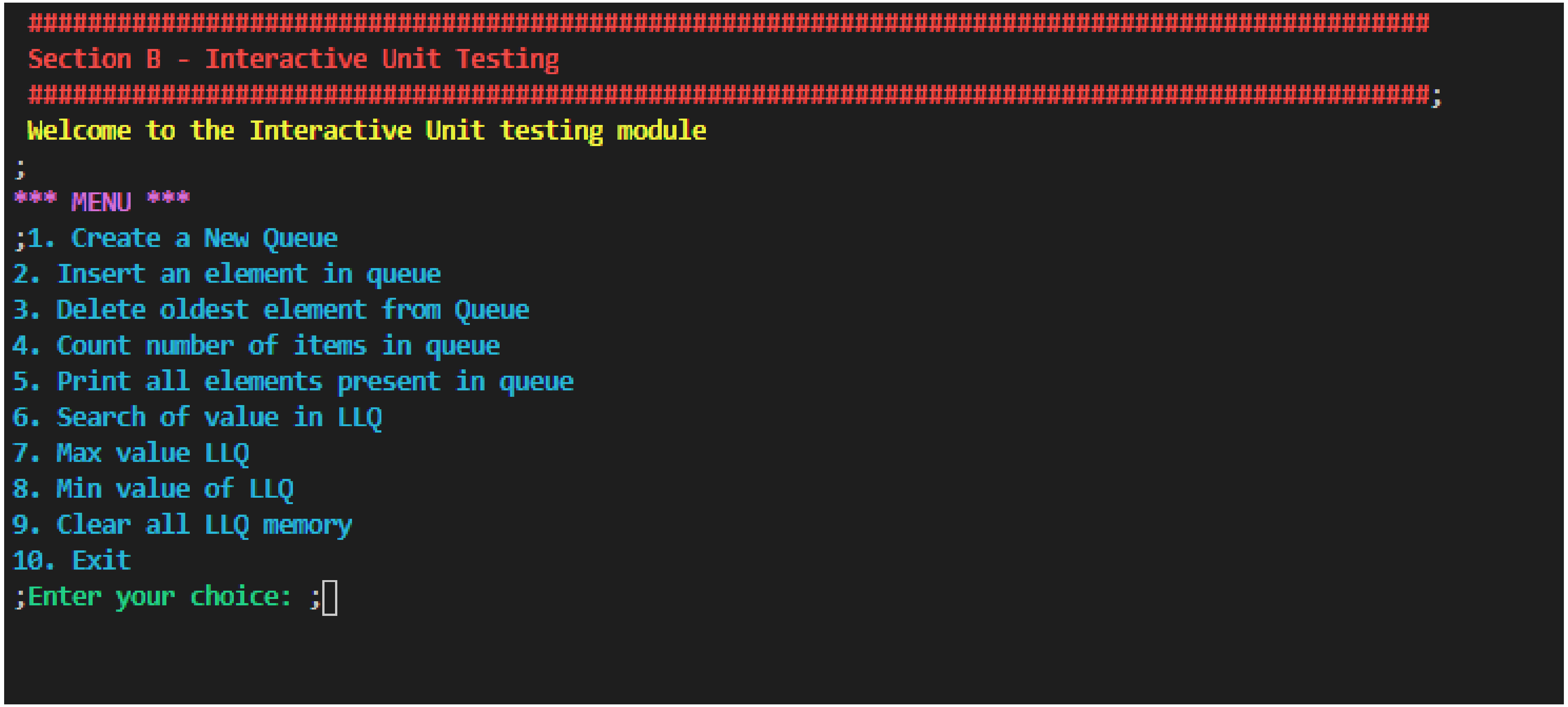


```
###############################################################################################
Section A - Automated Unit Testing
###############################################################################################
Welcome to the Automated Unit testing module
;
Case 1: Creating  an Empty LLQueue
;
 LLQueue is created!

 Total elements in LLQ is 0

 LLQ :
;
 Queue is Empty!!!
;;
 Case 2: Inserting the first element into an empty LLQ
;
Insertion is good
;
 LLQ :
;1.000000--->NULL
;
 Total items in LLQ  are 1.
```

Name: Amirtha Ganesh Pugazhendhi

2) Interactive Unit Testing module: This module is indented for the execution of unit test condition defined by the **user.(**i.e) Tester can input the test cases for the testing conditions. The creating **this interactive environment** was very challenging and important.

output



```
 ###############################################################################################
 Section B - Interactive Unit Testing
 ###############################################################################################;
 Welcome to the Interactive Unit testing module
;
*** MENU ***
;1. Create a New Queue
2. Insert an element in queue
3. Delete oldest element from Queue
4. Count number of items in queue
5. Print all elements present in queue
6. Search of value in LLQ
7. Max value LLQ
8. Min value of LLQ
9. Clear all LLQ memory
10. Exit
;Enter your choice: ;
```

Name: Amirtha Ganesh Pugazhendhi

Challenges faced:

1) Creating a function to search for an item present at the beginning, middle, and end of the linked list for a queue containing at least 4 elements.
    - This was quite challenging because it was complex it had 3 conditional flows namely:
        1) Checking whether the input element is present in the LLQueue and checking whether the element is **Head** of LLQueue
        2) Checking whether the input element is present in the LLQueue and checking whether the element is **Tail** of LLQueue
        3) Checking whether the input element is present in the LLQueue and checking whether the element is between Head and Tail (ie) **Middle** of the LLQueue

2) Implemented an additional “Interactive testing module” for better user interface . It was quite challenging.
    - This was also quite challenging because the unit test should be robust against randomness of the user inputs
    - It requires a creation of switch case structure for the evaluation of user choice of testing condition.
    - Implemented dynamic Text color graphics protocol for better readability and understanding