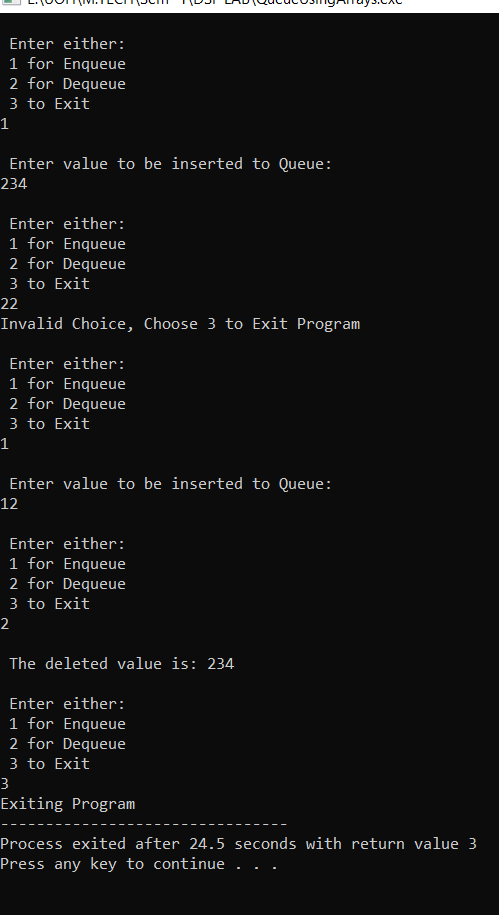
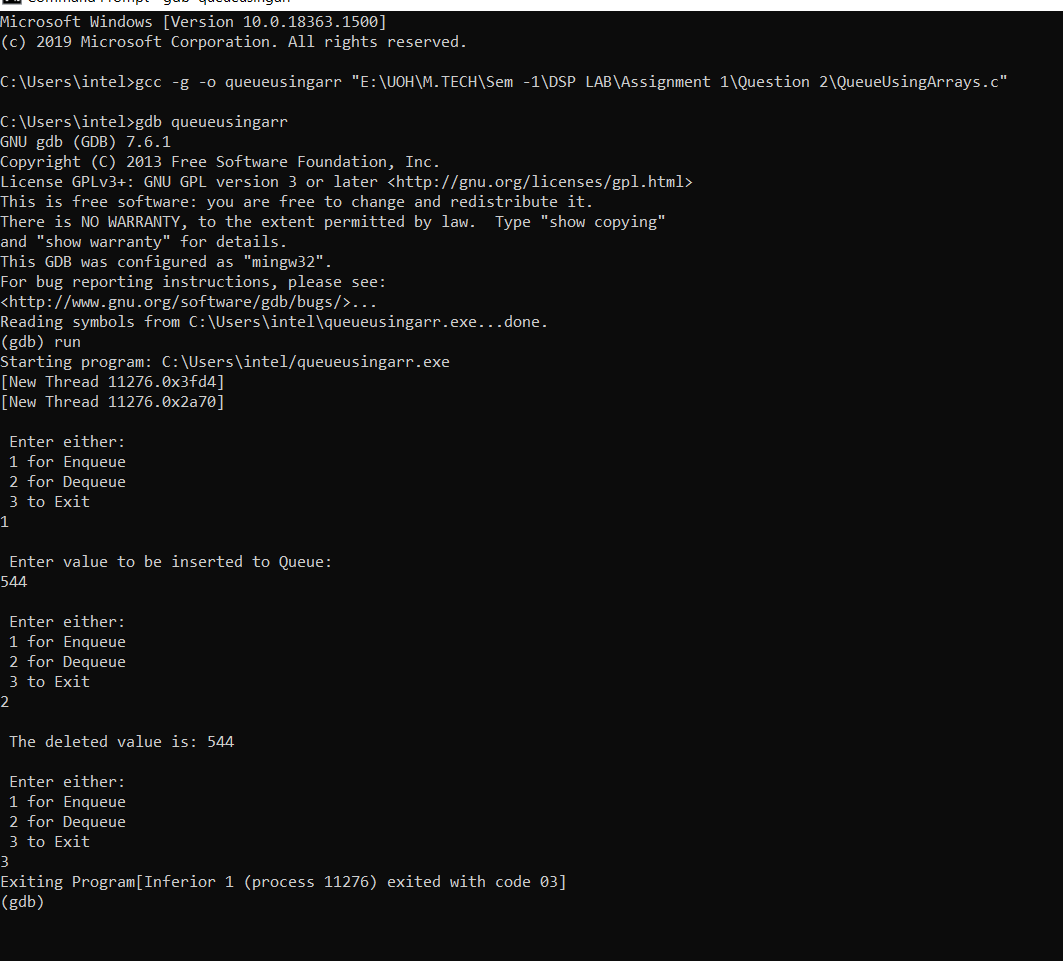
**2.1 Implement Queues using Arrays [Enqueue and Dequeue].**

Queue has been implemented using Arrays seperately.

* Seperate functions are created for enque and deque operations: addelement(), delelement()
* **addelement**(): pushes the input element in the array and rear is incremented by one.
* **delelement**(): is used to delete the first element entered by the user and front is incremented by one.
* Front keeps on incrementing as we delete all elements one by one, unless front becomes greater than rear, that would mean all elements have been deleted and queue / array is now empty
* Main(): function uses a switch case to help the user decide what operation they want to perform, among enqueue/dequeue/exit.

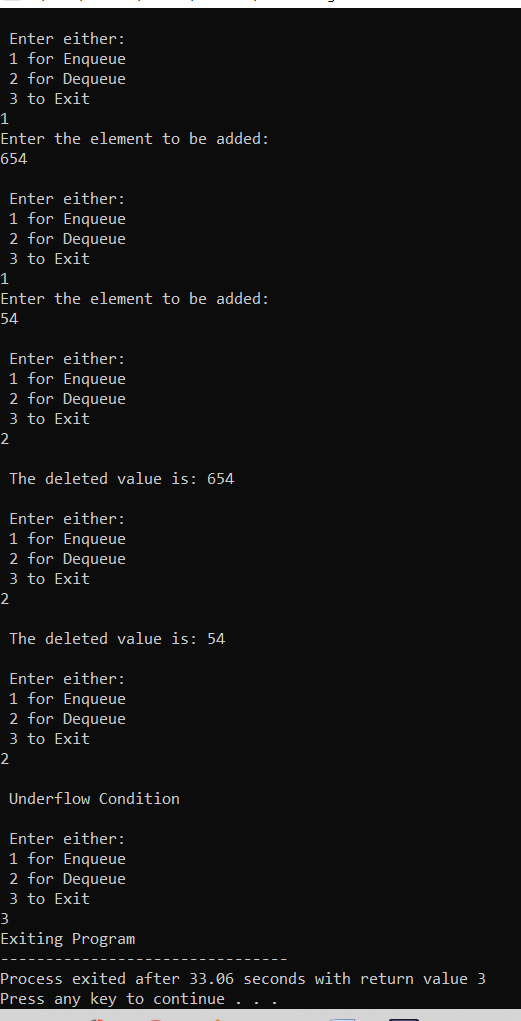
**Execution Screenshots:** ****

**Debug Screenshot:  
**

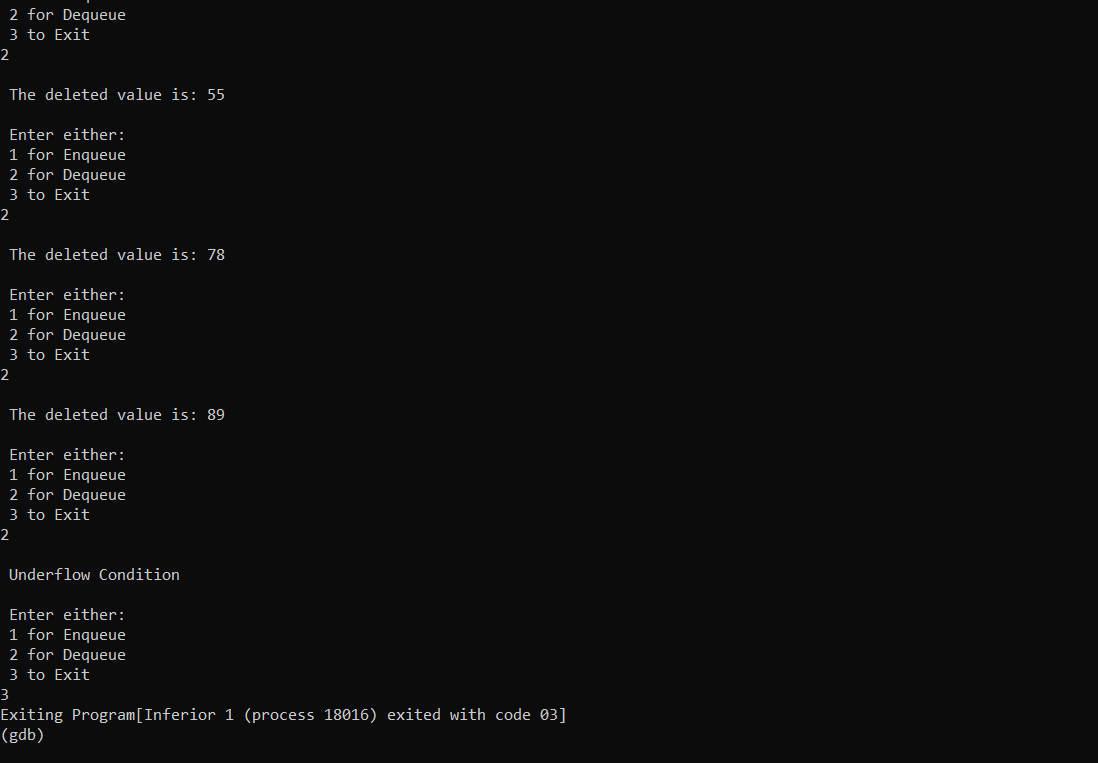
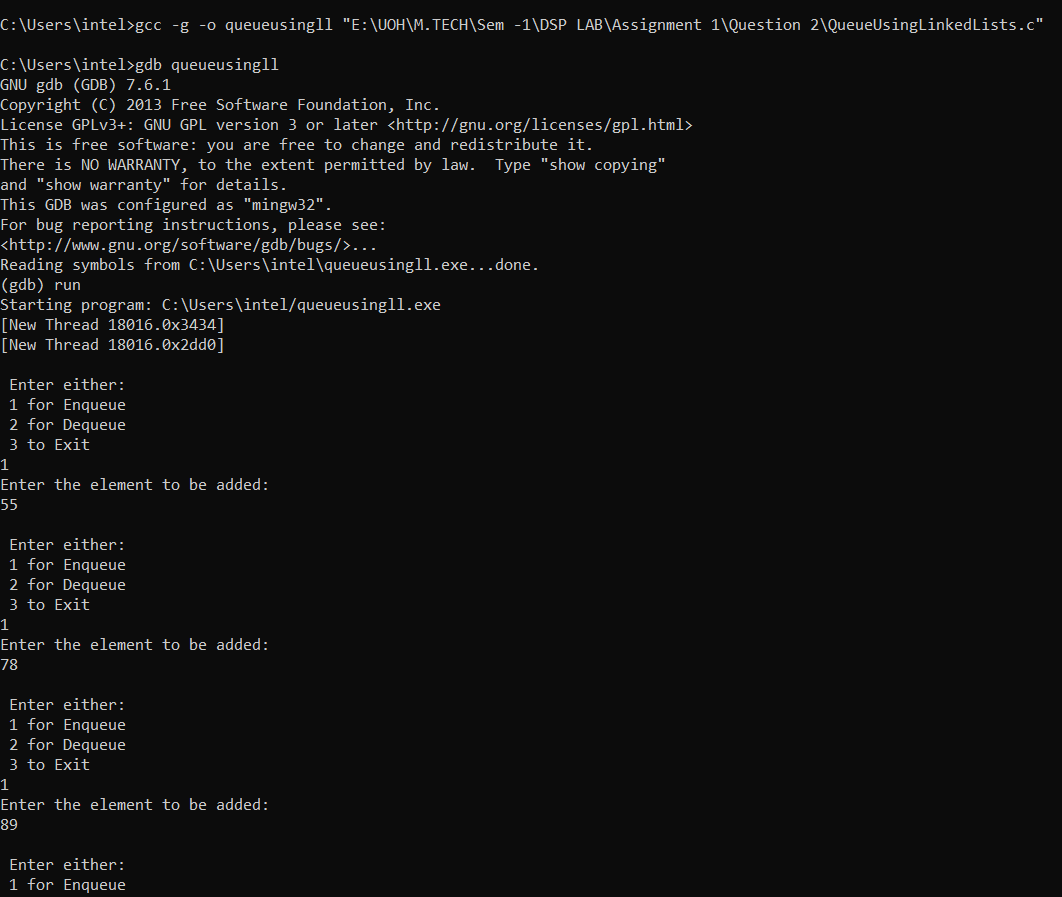
**2.2 Implement Queues using Linked Lists [Enqueue and Dequeue].**

Queue has been implemented using Linked Lists seperately.

* Seperate functions are created for enqueue and dequeue
* **addelement**(): pushes the input element a new node, connected to list and rear pointer is now pointing at the new node.
* **delelement**(): is used to delete the first element entered by the user and front pointer is incremented by one node
* Front keeps on incrementing as we delete all elements one by one, unless front becomes greater than rear, that would mean all elements have been deleted and queue / linkedlist is now empty
* Main(): function uses a switch case to help the user decide what operation they want to perform, among enqueue/dequeue/exit.

**Execution Screenshots:  
  
**

**Debug Screenshot:**

****