Name:

Wajiha

Roll no :

SU92-BSSEM-S24-040

Section :

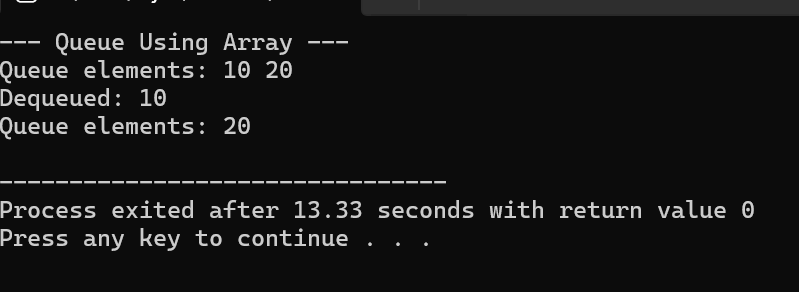
BSSE(3A)

Subject:

DSA(LAB)

(LAB No 11)

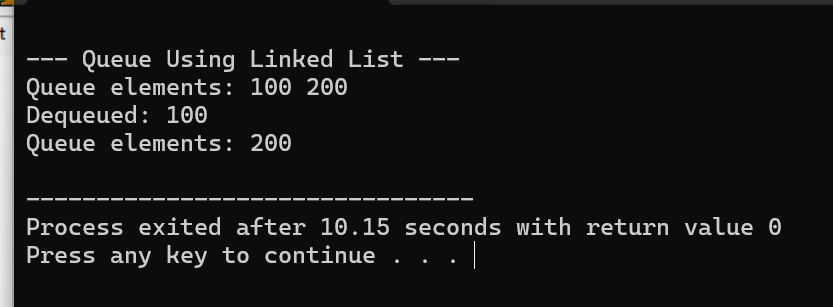
**✅ Queue with Array**

****

**Explanation:**

* **Structure:** Uses a fixed-size array of size 100.
* **Front & Rear:**
  + front: Points to the front of the queue (element to be dequeued next).
  + rear: Points to the rear of the queue (last inserted element).
* **Enqueue:**
  + Adds element at rear + 1
  + If queue is full (rear == MAX - 1), prints **Overflow**.
* **Dequeue:**
  + Removes element at front
  + If front > rear, queue is **empty**, prints **Underflow**.
* **Display:**
  + - Loops from front to rear, printing each element This implementation doesn’t reuse space from dequeued elements. To optimize, a circular queue can be used.

**✅ Queue with Linked List**

****

**Explanation:**

* **Structure:** Uses dynamically allocated nodes (Node) with two pointers:
  + front: Points to the first node.
  + rear: Points to the last node.
* **Enqueue:**
  + Creates a new node and adds it at the **end** of the list.
  + If the queue is empty, front and rear both point to the new node.
* **Dequeue:**
  + Deletes the node pointed by front.
  + Moves front to front->next.
  + If the queue becomes empty, also sets rear = nullptr.
* **Display:**
  + Traverses from front to rear printing each node’s data.
* This approach is **dynamic**, meaning it grows as needed — no limit unless memory is full.