**Lab 1:**

**Explaination:**

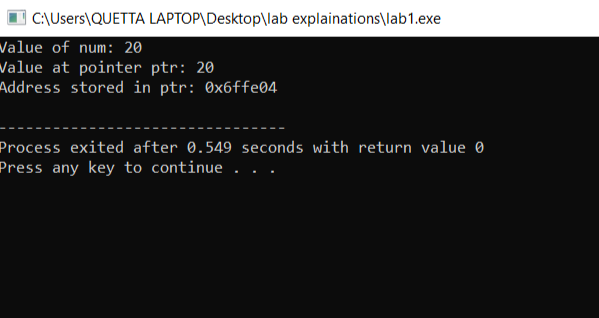
1. An integer num is declared and initialized with 10.

2. A pointer ptr is declared and assigned the address of num.

3. The value of num is modified to 20 using \*ptr.

4.The program prints:

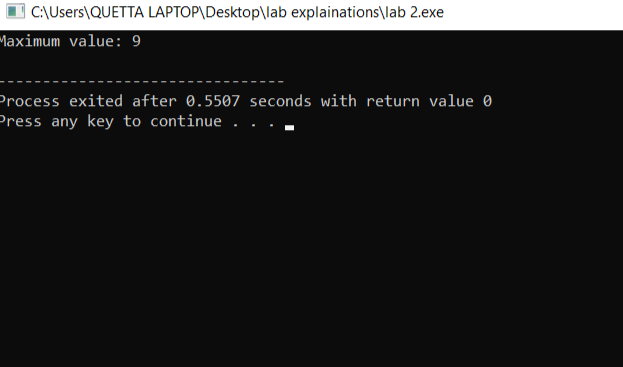
* The updated value of num.
* The value at \*ptr, confirming it reflects num.
* The memory address stored in ptr, verifying it points to num.



**Lab 2:**

**Explaination:**

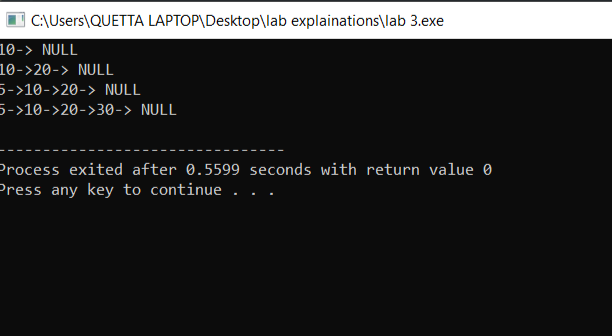
The function iterates through the array once, comparing each element to the current maximum. Since it performs a constant-time operation (if condition and assignment) for each element, the total number of operations is proportional to n. Therefore, the time complexity is **O(n)**.



**Lab 3:**

**Explanation:**

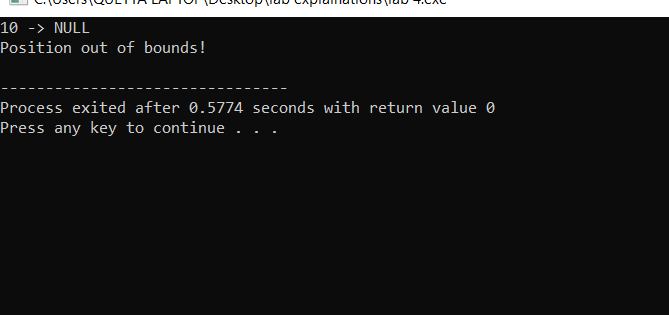
1. **Node Class:** Represents each element in the linked list, storing an integer data and a pointer next to the next node.
2. **LinkedList Class:** Manages the linked list with a head pointer.
   * **insertAtStart(int val):** Creates a new node, points it to the current head, and updates head to the new node.
   * **insertAtEnd(int val):** Creates a new node and traverses to the last node to link the new node at the end.
   * **display():** Iterates through the list and prints each node's value.
3. **Main Function:** Demonstrates insertions at the start and end, displaying the list after each operation.



**Lab 4:**

**Explanation:**

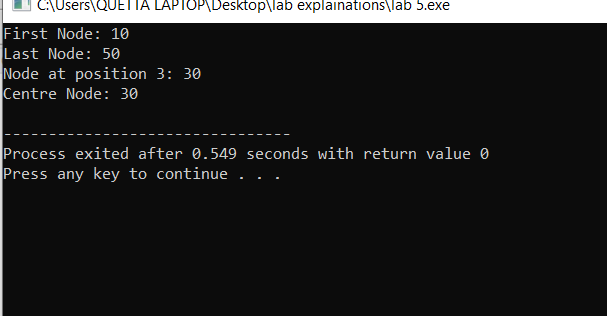
1. **Node Class:** Represents a node with data and next pointer.
2. **LinkedList Class:**
   * **insertAtPosition(int val, int pos):**
     + Handles invalid positions (pos < 1).
     + If inserting at position 1, updates head.
     + Otherwise, traverses to pos - 1 and inserts the node.
     + If position exceeds the list length, it prints an error.
   * **display():** Prints the linked list.
3. **Main Function:** Inserts nodes at different positions and demonstrates position validation.



**Lab 5:**

**Explanation:**

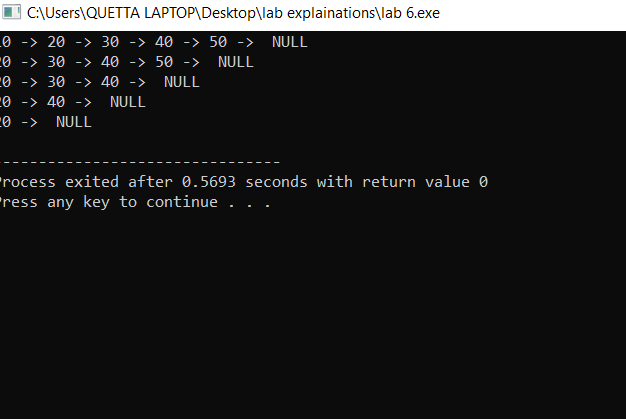
1. **Node Class:** Represents a node with data and next pointer.
2. **LinkedList Class:**
   * **insertAtEnd(int val):** Inserts a node at the end.
   * **displayFirstNode():** Prints the first node (head).
   * **displayLastNode():** Traverses to and prints the last node.
   * **displayNthNode(int n):** Finds and prints the Nth node.
   * **displayCentreNode():** Uses slow and fast pointers to find the middle node.
3. **Main Function:** Uses n as the object of LinkedList instead of list.



**Lab 6:**

### Explanation:

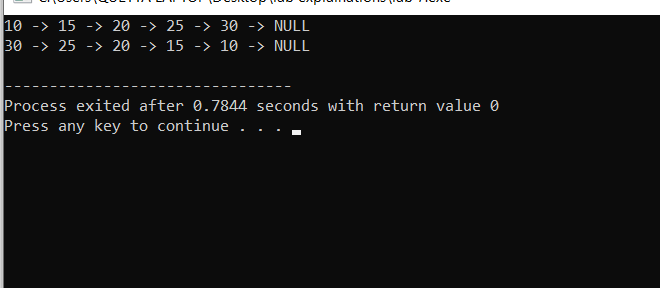
1. **Node Class:** Represents each node with data and next pointer.
2. **LinkedList Class:**
   * **insertAtEnd(int val):** Inserts a node at the end.
   * **deleteFirstNode():** Deletes the first node.
   * **deleteLastNode():** Deletes the last node.
   * **deleteNthNode(int n):** Deletes the node at position n, ensuring valid position handling.
   * **deleteCentreNode():** Uses slow and fast pointers to find and delete the middle node.
   * **displayList():** Prints the linked list.
3. **Main Function:** Demonstrates the deletion functions on a sample linked list.

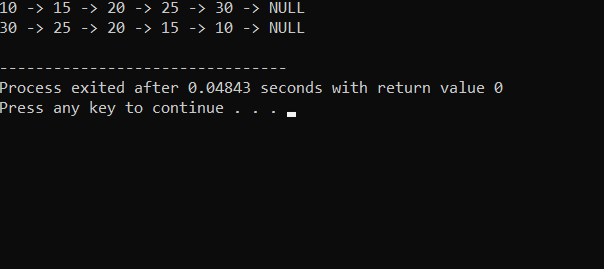


**Lab 7:**

### Explanation:

* **Node Class:** Represents a singly linked list node.
* **LinkedList Class:**
  + insertAtFirst(int val): Inserts at the beginning.
  + insertAtLast(int val): Inserts at the end.
  + insertAtNth(int n, int val): Inserts at the nth position.
  + insertAtCentre(int val): Inserts in the middle using slow and fast pointers.
  + displayInOrder(): Displays the list in order.
* **Main Function:** Demonstrates insertions and prints the list.

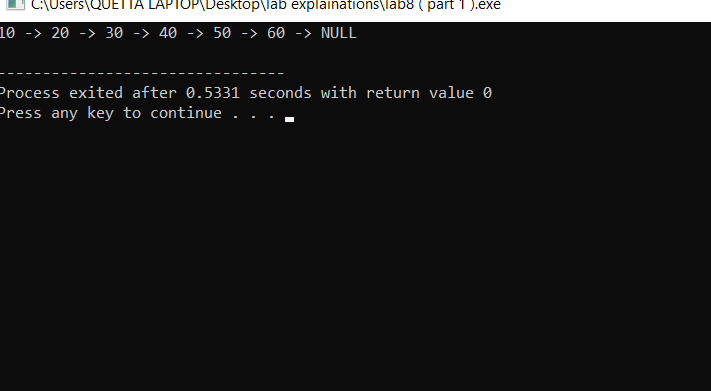




**Lab 8(singly list merging):**

### Explanation:

* **Node Class:** Represents a node in a singly linked list.
* **LinkedList Class:**
  + insertAtLast(int val): Inserts at the end.
  + mergeList(LinkedList& other): Merges another list into the current list.
  + displayList(): Displays the list.
* **Main Function:** Creates two lists, merges them, and displays the result.



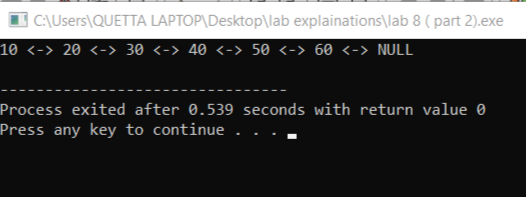
**Lab 8(doubly list merging):**

 **DNode Class:** Represents a node in a doubly linked list.

 **DoublyLinkedList Class:**

* insertAtLast(int val): Inserts at the end.
* mergeList(DoublyLinkedList& other): Merges another list into the current list.
* displayList(): Displays the list.

 **Main Function:** Creates two lists, merges them, and displays the result.

****

**Lab 7:**

### Explanation:

* **Node Class:** Represents a node in a circular linked list.
* **CircularLinkedList Class:**
  + insertAtFirst(int val): Inserts at the beginning.
  + insertAtLast(int val): Inserts at the end.
  + insertAtNth(int n, int val): Inserts at the nth position.
  + insertAtCentre(int val): Inserts in the middle using slow and fast pointers.
  + displayInOrder(): Displays the list in order.
  + displayReverse(): Displays the list in reverse using recursion.
* **Main Function:** Demonstrates insertions and prints the list in both orders.

