

**lab tasks**

**NAME:**

**Tayyaba Muslim**

**ROLL NO:**

**SU92-BSSEM-S24-076**

**SECTION:**

**BSSE-3A**

**COURSE:**

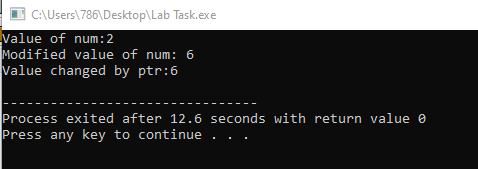
**DSA LAB**

**SUBMITTED TO:**

**Sir Rasik Ali**

**Task 01:**

**Screen shot:**

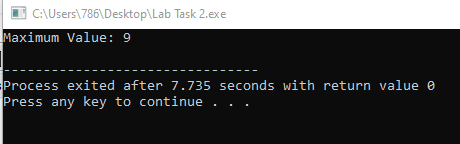


**Explanation:**

We declare an integer variable num and a pointer that holds the address of number. Modifie the value of a number by dereferencing the pointer.

**Task 02:**

**Screen shot:**

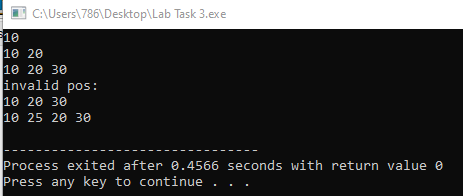
****

**Explanation:**

* We iterate through the array once to find the maximum value.
* The time complexity is O(n) because we visit each element exactly once in the array.

**Task 03,4:**

**Screen shot:**

****

**Explanation:**

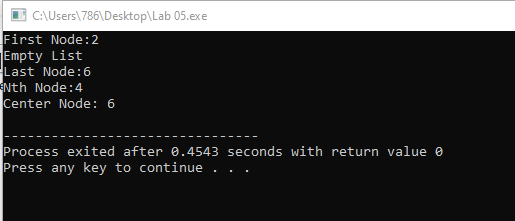
insertAtStart() inserts a new node at the beginning.

insertAtEnd() inserts a new node at the end.

The insertAtPosition() function inserts a new node at a given position.

**Task 05:**

**Screen shot:**

****

**Explanation:**

 **First Node**: Display the head node.

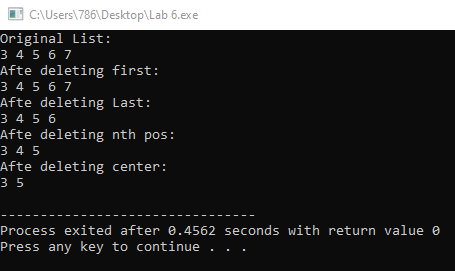
 **Last Node**: Traverse to the end of the list and display the last node.

 **Nth Node**: Traverse to the Nth node and display its value.

 **Center Node**: Use the slow and fast pointer technique to find the middle of the list.

**Task 06:**

**Screen shot:**

****

**Explanation:**

 **Delete the first node**: The head of the list is updated to the next node.

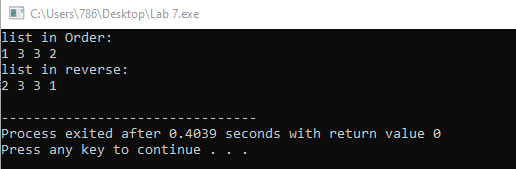
 **Delete the last node**: Traverse to the second last node, update its next pointer to nullptr, and delete the last node.

 **Delete the Nth node**: Traverse to the Nth node and delete it.

 **Delete the center node**: Use the slow and fast pointer technique to find and delete the center node.

**Task 07:**

**Screen shot:**

****

**Explanation:**

** Insert at first**: Create a new node, set its next to the current head, and update the head.

 **Insert at last**: Traverse to the last node, then append the new node to it.

 **Insert at Nth location**: Traverse to the Nth position and insert the node at that position.

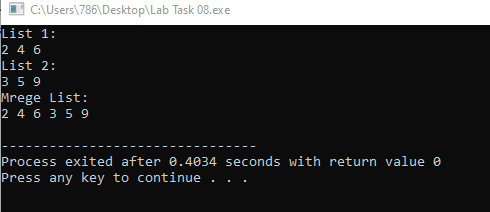
 **Insert at center**: Use the slow and fast pointer technique to find the center and insert the node there.

 **Display in order**: Traverse from head to tail.

 **Display in reverse**: Traverse from the tail to the head using the prev pointers.

**Task 08:**

**Screen shot:**

****

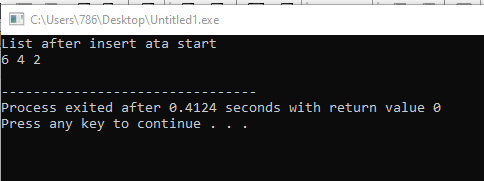
**Explanation:**

 **Singly Linked Lists**: Iterate through both lists, append the second list to the first list, and display the merged list.

 **Doubly Linked Lists**: Similarly, traverse both lists and merge them by updating the next and prev pointers.

**Task 09:**

**Screen shot:**

****

**Explanation:**

 **Circular Linked List**: This is a type of linked list where the last node points back to the first node, making it circular.

 **Insert at Start**: This method inserts a new node at the beginning of the list. In a circular linked list, this involves pointing the new node's next to the previous head and updating the last node's next to the new node.

 **Display List**: This method traverses the circular linked list and prints the data of each node until it loops back to the head.