**Ahmad Hussain:**

**Roll no**. BSSEM-64

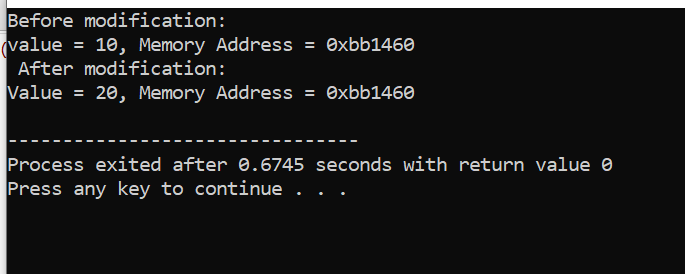
**Section**: 3A.

**LAB 1:**

**Explanation:**

This C++ program demonstrates how pointers work with variables. It declares an integer num and a pointer ptr that stores its address. Initially, the program prints the value of num, the address stored in ptr, and the value at that address using \*ptr. Then, it modifies num through \*ptr by assigning it 20. After modification, it prints the updated value of num, confirming that changing \*ptr directly updates num since ptr points to its memory location.

**Output:**

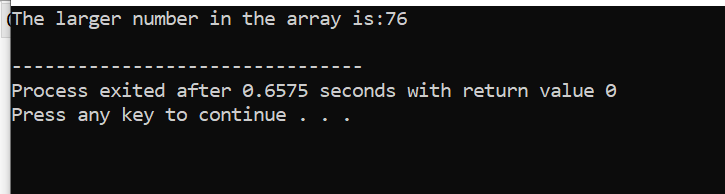


**LAB 2:**

**Explantion:**

This C++ program finds the maximum value in an array. The findMax function takes an array and its size as input. It first checks if the array is empty; if so, it returns -1. Otherwise, it initializes maxVal with the first element and iterates through the array, updating maxVal if a larger value is found. In main(), an integer array is defined, and its size is calculated. The function findMax is called, and the largest number is printed.

**Output:**

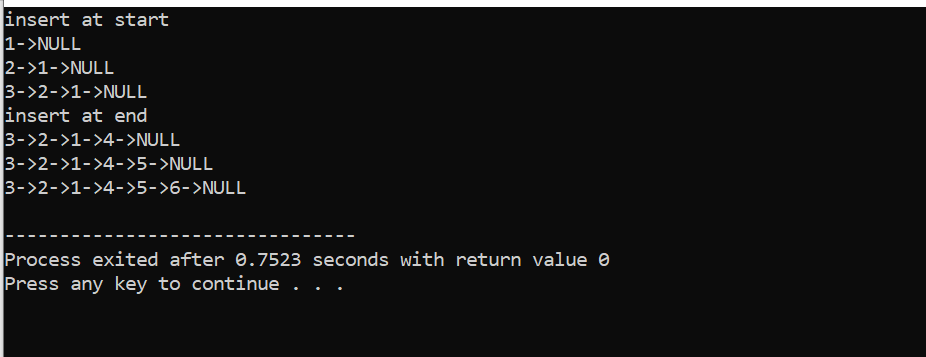
****

**LAB 3:**

**Explanation:**

This C++ program implements a simple singly linked list with insert operations at the start and end. The Node class stores data and a pointer to the next node. The LinkedList class manages the list with a head pointer. The insertATstart function adds a node at the beginning, while insertATend adds a node at the end by traversing to the last node. The display function prints the list. In main(), nodes are inserted at both start and end, displaying the updated list after each insertion.

**Output:**

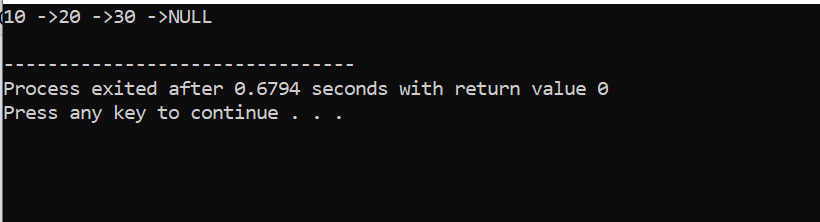
****

**LAB 4:**

**Explanation:**

This C++ program implements a singly linked list with an insert at a specific position feature. The Node class represents each node with data and a next pointer. The LinkedList class manages the list, and the insertATpos function inserts a node at a given position. It checks if the position is valid, inserts at the head if pos == 1, or traverses to the correct position before inserting. The display function prints the list. In main(), nodes are inserted at positions 1, 2, and 3, and the final list is displayed.

**Output:**

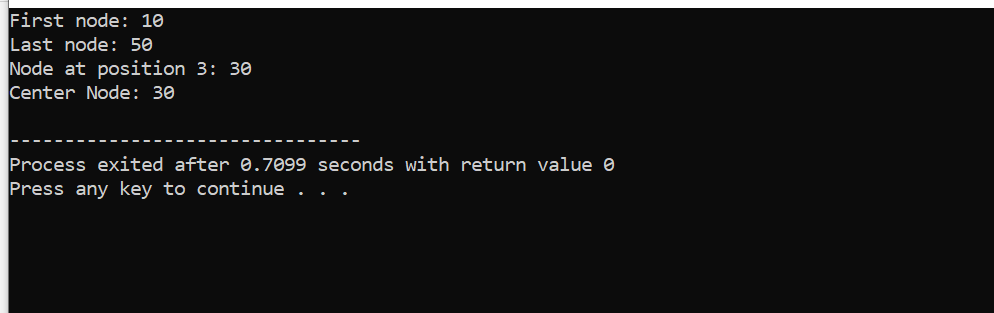
****

**LAB 5:**

**Explanation:**

This C++ program implements a singly linked list with functions to display the first, last, Nth, and center nodes. The Node class represents each node, while the LinkedList class manages the list. Nodes are added at the end using insertAtEnd(). The displayFirstNode(), displaylastNode(), displayNthNode(n), and displayCenterNode() functions retrieve and print the respective nodes. The main() function inserts five nodes (10, 20, 30, 40, 50), displays the list, and then calls each function to show the required node details.

**Output:**

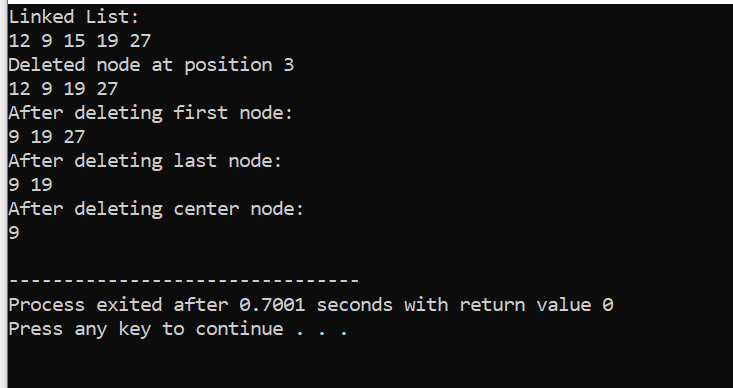
****

**LAB 6:**

**Explanation:**

This C++ program implements a singly linked list with functions to delete the first, last, Nth, and center nodes. The insert\_at\_end() function adds nodes at the end, and display() prints the list. The deletefirstnode(), deletelastnode(), deleteNthnode(n), and deleteCenternode() functions remove the respective nodes while handling edge cases (empty list, single node). In main(), a list is created with values 10, 20, 30, 40, 50, and deletion functions are called sequentially, displaying the list after each operation.

**Output:**

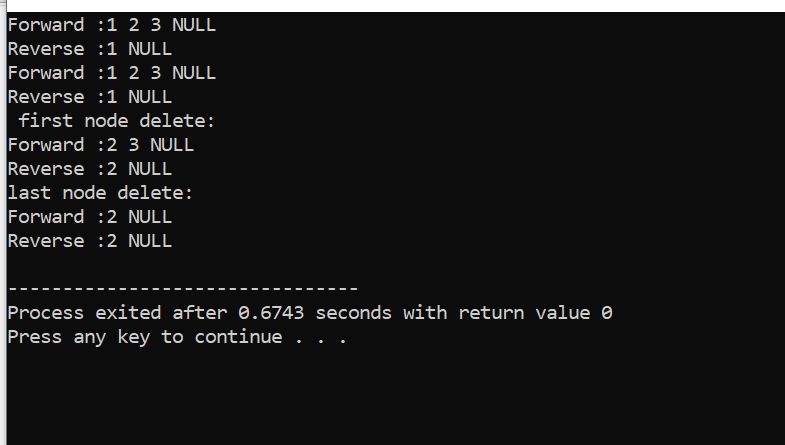
****

**LAB 7:**

**Explanation:**

This program implements a doubly linked list with functions to insert and delete nodes at the start and end while maintaining both forward and backward links. The insertAtstart() and insertAtend() functions add nodes at the beginning and end, respectively. The deleteAtstart() and deleteAtend() functions remove nodes while handling edge cases (empty list, single node). The displayForward() prints the list from head to tail, while displayReverse() prints from tail to head. In main(), nodes are inserted, displayed in both directions, deleted, and displayed again to show change**s.**

**Output:**

****

**LAB 8:**

**Explanation:**

This program merges two singly linked lists and doubly linked lists and displays them in both forward and reverse order.

Singly Linked List:

Two linked lists (list1 and list2) are created, and nodes are inserted.

The merge() function connects the last node of list1 to the head of list2.

After merging, the combined list is displayed.

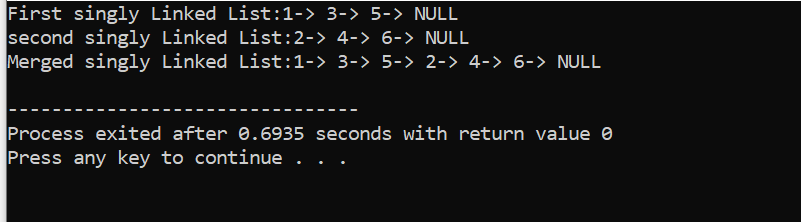
Doubly Linked List:

Similar to the singly linked list but with prev and next pointers.

The merge() function links list1's tail to list2's head and updates pointers correctly.

The merged list is displayed in both forward and reverse order.

**Output:**

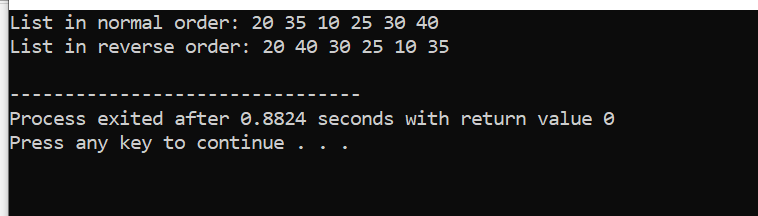
****

**LAB 9:**

**Explanation:**

This program implements a circular doubly linked list where nodes are connected in a loop, with the last node pointing back to the first. It includes functions to insert a node at the beginning, end, Nth position, and center of the list. The displayForward() and displayReverse() functions print the list in both directions. In main(), nodes are inserted at various positions, and the list is displayed in both forward and reverse order.

**Output:**

****