EX.NO : 6(A)	SINGLY LINKED LIST
DATE :	

PROGRAM STATEMENT :

To write a CPP program to INSERT an Element at location 2 using STL and Display the same.

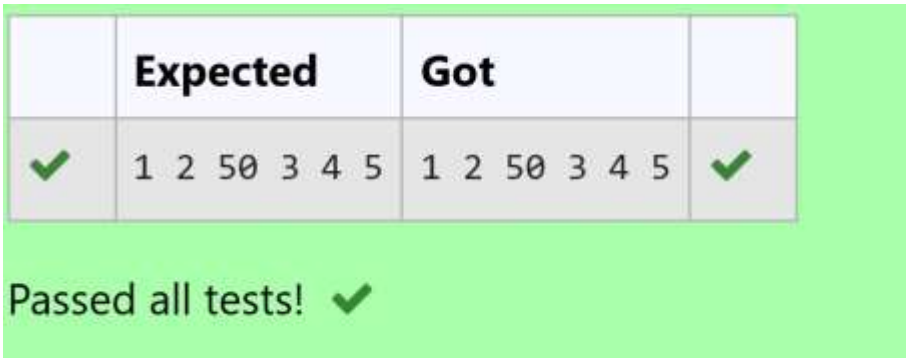
ALGORITHM:

1. Start the program
2. Initialize a forward_list of integers, flist, with values {1, 2, 3, 4, 5}.
3. Create an iterator it pointing to the beginning of the list.
4. Move the iterator it to the second position in the list.
5. Insert the integer 50 after the position indicated by it.
6. Use a loop to iterate over flist from the beginning to the end, printing each element.
7. End the program.

PROGRAM:

```
#include<iostream>
 #include<forward_list>
 using namespace std;
 int main()
{
 forward_list<int>flist;
flist={1, 2, 3, 4, 5};
auto it=flist.begin();
it++;
   flist.insert_after(it,50);
   for(it=flist.begin();it!=flist.end();it++){
      cout<<*it<<"";
   }
}
```

OUTPUT :



	Expected	Got	
✔	1 2 50 3 4 5	1 2 50 3 4 5	✔

Passed all tests! ✔

RESULT :

Thus, the C++ program to INSERT an Element at location 2 using STL and Display the same is created successfully.

EX.NO : 6(B)	DOUBLY LINKED LIST
DATE :	

PROGRAM STATEMENT:

To write a CPP program to INSERT an Element at LOCATION 1 in Doubly Linked List Using STL and Display the same.

ALGORITHM:

1. Start the program.
2. Initialize an empty list ele of integers.
3. Use a loop to input 5 integers from the user, adding each to the end of ele using push_back.
4. Input an additional integer, ent, and add it to the front of ele using push_front.
5. Print "List: ".
6. Use a loop to iterate through ele from beginning to end, printing each element.
7. End the program.

PROGRAM:

```
#include<iostream>
#include<list>
usingnamespacestd;
int main()
{ list<int> ele; int
   num,i,ent;
   for(i=0;i<5;i++
   )
   {
      cin>>num;
      ele.push_back(num);
   } cin>>ent;
   ele.push_front(ent);
   cout<<"List: ";
   for(autoit=ele.begin();it!=ele.end();it++)
   { cout<<*it<<" ";
   }

}
```

OUTPUT :

	Input	Expected	Got	
✔	5 6 7 8 9 2	List: 2 5 6 7 8 9	List: 2 5 6 7 8 9	✔
✔	10 20 30 40 50 230	List: 230 10 20 30 40 50	List: 230 10 20 30 40 50	✔
✔	12 23 34 45 56 67	List: 67 12 23 34 45 56	List: 67 12 23 34 45 56	✔

Passed all tests! ✔

RESULT:

Thus, the C++ program to INSERT an Element at LOCATION 1 in Doubly Linked List Using STL and Display the same is created successfully.

EX.NO : 6(C)	CIRCULAR LINKED LIST
DATE:	

PROGRAM STATEMENT:

To write a CPP program to SEARCH an element from the Circularly Linked List and Display the same.

ALGORITHM:

1. Start the program.
2. Define a Node class with data and nextptr (pointer to next node).
3. Create a create() function to create a new node with given data and set its nextptr to 0.
4. If the list is empty (head == 0), set both head and tail to the new node, otherwise, add the node at the end and update tail->nextptr to head.
5. Implement the display() function to traverse the circular linked list and print each node's data until it reaches back to head.
6. Implement the search() function to traverse the list and compare each node's data with a given target, printing whether it is found or not.
7. In main(), input 5 integers to create the list, input a target element, display the list, and search for the target in the list. 8. End the program

PROGRAM:

```
#include <iostream>
using namespace std;

// Node structure
struct Node {
   int data;
   Node* next;
};

// Class for Circular Linked List
class CircularLinkedList {
private:
   Node* last;  // Points to the last node in the circular list

public:
   CircularLinkedList() {
      last = nullptr;
```

```
}

// Function to insert node at the end
void insertEnd(int value) {
    Node* newNode = new Node();
    newNode->data = value;

    if (last == nullptr) {
        newNode->next = newNode;
        last = newNode;
    } else {
        newNode->next = last->next;
        last->next = newNode;
        last = newNode;
    }
}

// Function to search an element in the CLL
bool search(int key) {
    if (last == nullptr) {
        cout << "List is empty.\n";
        return false;
    }

    Node* temp = last->next;
    do {
        if (temp->data == key) {
            cout << "Element " << key << " found in the list.\n";
            return true;
        }
        temp = temp->next;
    } while (temp != last->next);

    cout << "Element " << key << " not found in the list.\n";
    return false;
}

// Display the list
void display() {
    if (last == nullptr) {
        cout << "List is empty.\n";
        return;
    }

    Node* temp = last->next;
    cout << "Circular Linked List: ";
```

```
        do {
            cout << temp->data << " ";
            temp = temp->next;
        } while (temp != last->next);
        cout << endl;
    }
};

int main() {
    CircularLinkedList cll;

    // Insert elements
    cll.insertEnd(10);
    cll.insertEnd(20);
    cll.insertEnd(30);
    cll.insertEnd(40);

    // Display the list
    cll.display();

    // Search for elements
    cll.search(30);  // Present
    cll.search(50);  // Not present

    return 0;
```

OUTPUT :

	Input	Expected	Got	
✔	10 20 30 40 50 40	Data = 10 Data = 20 Data = 30 Data = 40 Data = 50 Element 40 Found	Data = 10 Data = 20 Data = 30 Data = 40 Data = 50 Element 40 Found	✔
✔	12 23 56 34 78 34	Data = 12 Data = 23 Data = 56 Data = 34 Data = 78 Element 34 Found	Data = 12 Data = 23 Data = 56 Data = 34 Data = 78 Element 34 Found	✔
✔	100 200 300 400 500 400	Data = 100 Data = 200 Data = 300 Data = 400 Data = 500 Element 400 Found	Data = 100 Data = 200 Data = 300 Data = 400 Data = 500 Element 400 Found	✔
✔	10 20 30 40 50 75	Data = 10 Data = 20 Data = 30 Data = 40 Data = 50 Element not Found	Data = 10 Data = 20 Data = 30 Data = 40 Data = 50 Element not Found	✔

Passed all tests! ✔

RESULT:

Thus, the C++ program to SEARCH an element from the Circularly Linked List and Display the same is created successfully.

EX.NO : 6(D)	POLYNOMIAL MANIPULATION
DATE :	

PROGRAM STATEMENT:

To debug a C++ function int printList(Node *head) to print the polynomial expression in polynomial addition program using Linked list concept.

ALGORITHM:

1. Define a function printList that takes a pointer to the head node of a linked list as input.
2. Print "Linked List" as the header.
3. Use a while loop to traverse the linked list as long as head is not NULL:
4. For each node, print its coeff (coefficient) followed by "x^" and its power (exponent).
5. Move to the next node by setting head = head->next.
6. Once all nodes are printed, return 1 to indicate successful completion.
7. End the program.

PROGRAM:

```
int printList(Node*head)
{ cout<<"Linked List"<<endl; while(head!=NULL){
   cout<<""<<head->coeff<<"x^"<<head->power;
  head=head->next;
   } return 1;
}
```

OUTPUT :

	Input	Expected	Got	
✔	-	Linked List 10x^3 15x^1 Linked List 6x^3 4x^1 Addition: 16x^3 19x^1	Linked List 10x^3 15x^1 Linked List 6x^3 4x^1 Addition: 16x^3 19x^1	✔

Passed all tests! ✔

RESULT:

Thus, the C++ program to print the polynomial expression in polynomial addition program using Linked list concept is created successfully.