DATA STRUCTURES LABORATORY MANUAL – ICE 2144

III SEMESTER B. TECH

EXPT. 6 LINKED LIST

1) Write a program to create and insert a node at the beginning of a list. Display the values of the list.

(i) Creation:
Step 1 - Define a Node structure with two members data and next
Step 2 - Define a Node pointer 'head' and set it to NULL.

(ii) Insertion:
In a single linked list, the insertion operation can be performed in three ways. They are as follows: Inserting at Beginning of the list, Inserting at End of the list and Inserting at Specific location in the list.

a) Inserting at Beginning of the list:
Step 1 - Create a newNode with given value.
Step 2 - Check whether list is Empty (head == NULL)
Step 3 - If it is Empty then, set newNode→next = NULL and head = newNode.
Step 4 - If it is Not Empty then, set newNode→next = head and head = newNode.

(iii) Display:
Step 1 - Check whether list is Empty (head == NULL)
Step 2 - If it is Empty then, display 'List is Empty!!!' and terminate the function.
Step 3 - If it is Not Empty then, define a Node pointer 'temp' and initialize with head.
Step 4 - Keep displaying temp → data with an arrow (→) until temp reaches to the last node Step 5 - Finally display temp → data with arrow pointing to NULL (temp → data ---> NULL).

Program:

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

class Node                  //Creating a node
{
        public:
                int data;
                Node* next;
};
void push(Node** head_ref, int new_data)        // inserts a new node on the
                                                // front of the list.
{
        // 1. allocate node
        Node* new_node = new Node();
        // 2. put in the data
        new_node->data = new_data;
        // 3. Make next of new node as head
        new_node->next = (*head_ref);
        // 4. Move the head to point to the new node
        (*head_ref) = new_node;
```

```
}
void printlist(Node *node)
{
        while(node!=NULL)
        {
                cout<<""<<node->data;
                node = node->next;
        }
}
int main()
{
        Node *head = 0;
        push(&head, 7);
        push(&head, 1);
        push(&head, 8);
        push(&head, 4);
        cout<<"Created list is:";
        printlist(head);
        return 0;
}
```

2) Write a function that searches a given key ‘x’ in a given singly linked list.

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

// Link list node
class Node
{
    public:
    int key;
    Node* next;
};

/* Given a reference (pointer to pointer) to the head of a list and an int, push a new node on the front of the list. */
void push(Node** head_ref, int new_key)
{
    // Allocate node
    Node* new_node = new Node();

    // Put in the key
    new_node->key = new_key;

    // Link the old list of the new node
    new_node->next = (*head_ref);

    // Move the head to point to the new node
    (*head_ref) = new_node;
}
```

```
// Checks whether the value x is present in linked list
bool search(Node* head, int x)
{
        Node* current = head;
        while (current != NULL)
        {
           if (current->key == x)
              return true;
           current = current->next;
        }
        return false;
}

int main()
{
        // Start with the empty list
        Node* head = NULL;
        int x = 21;
         // Use push() to construct list 14->21->11->30->10
        push(&head, 10);
        push(&head, 30);
        push(&head, 11);
        push(&head, 21);
        push(&head, 14);

        search(head, 21)? cout<<"Yes" : cout<<"No";
        return 0;
}
```

3) Write a program to delete the last node of the list.

Deletion:

In a single linked list, the deletion operation can be performed in three ways. They are as follows: Deleting at Beginning of the list, Deleting at End of the list and Deleting at Specific location in the list.

a) Deleting at End of the list:

Step1: Traverse link list to second last element
Step2: Change its next pointer to null
Step3: Free the memory of the last node.

Program:

```
#include<iostream>
using namespace std;
class node
{
        public:
                int data;
                node*next;
                node(int d)
```

```
		{
			data=d;
			node*next= NULL;
		}
};
void insertAtFirst(node*&head, int data)
{
	node*n= new node(data);
	n->next= head;
	head=n;
}
void printNode(node*head)
{
	while(head!=NULL)
	{
		cout<<head->data<<"->";
		head=head->next;
	}
	cout<<endl;
}
void deleteatTail(node*head)
{
	node*prev= NULL;
	node*temp= head;
	while(temp->next!=NULL)
	{
		prev= temp;
		temp=temp->next;
	}
	delete temp;
	prev->next= NULL;
	return;
}
int main()
{
	node*head= NULL;
	insertAtFirst(head,5);
	insertAtFirst(head,4);
	insertAtFirst(head,3);
	insertAtFirst(head,2);
	insertAtFirst(head,1);
	printNode(head);
	deleteatTail(head);
	printNode(head);
	return 0;
}
```

DATA STRUCTURES LABORATORY MANUAL – ICE 2144

III SEMESTER B. TECH

4) Write a program to insert an element in the front of a doubly linked list.

Step1: Firstly, allocate a new node.
Step2: Now put the required data in the new node.
Step3: Make the next of new node point to the current head of the doubly linked list.
Step4: Make the previous of the current head point to new node.
Step5: Lastly, point head to new node.

Program:

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

class Node
{
        public:
        int data;
        Node* next;
        Node* prev;
};

class LinkedList
{
        private:
                Node* head;
        public:
                LinkedList()
                {
                        head = NULL;
                }

   //Add new element at the start of the list
   void push_front(int newElement)
        {
                Node* newNode = new Node();
                newNode->data = newElement;
                newNode->next = NULL;
                newNode->prev = NULL;
                if(head == NULL)
                {
                        head = newNode;
                }
                else
                {
                        head->prev = newNode;
                        newNode->next = head;
                        head = newNode;
                }
        }
```

```
    //display the content of the list
    void PrintList()
        {
         Node* temp = head;
        if(temp != NULL)
                {
                        cout<<"The list contains: ";
                        while(temp != NULL)
                        {
                                cout<<temp->data<<" ";
                                temp = temp->next;
                        }
                        cout<<endl;
                }
        else
                {
                        cout<<"The list is empty.\n";
                }
        }
};

// test the code
int main()
{
        LinkedList MyList;

        //Add three elements at the start of the list.
        MyList.push_front(10);
        MyList.push_front(20);
        MyList.push_front(30);
        MyList.PrintList();

        return 0;
}
```

Exercise:

1) Write a program to insert a node at the end of the linked list. Also perform the delete operation at a given position in the same list.
2) Write a program to count the number of nodes in a singly linked list using a function.
3) Write a program to delete the prime numbers from a single linked list. Display the list before and after deletion.
4) Write a program the delete all the even nodes from a doubly linked list.