Jaypee University of Engineering and Technology, Guna

[LAB ACTIVITY 7]

DATA STRUCTURES LAB

18B17CI371

Name: Palash Mishra

Enroll No. : 201B172

```
/*Write a function Insert_Beginning() to insert a new node at the
beginning of singly linked list. Call this function N time to
create a linked list with N nodes.
Also write display function to print the linked list.*/
#include<iostream>
using namespace std;
struct node
{
    int data;
    struct node *next;
};
struct node*head=NULL;
void Insert_Beginning(int val)
{
    struct node *newNode = new struct node;
    newNode->data=val;
    newNode->next=head;
    head=newNode;
}
void Print_List()
{
    struct node *temp=head;
      while(temp != NULL)
    {
         cout<<temp->data<<" ";
         temp = temp->next;
    }
}
int main()
{
    int N;
    cout<<"Enter : How many time you want to insert at begining :
";
    cin>>N;
    int value;
    cout<<"\nEnter the Data values : ";
    for(int i=1;i<=N;i++)
    {
        cin>>value;
        Insert_Beginning(value);
    }
    Print_List();
    return 0;
}
/*Write a menu driven program using switch-case to insert the node
at beginning,
```

```
after specified position and at the end of linked list.*/
#include <iostream>
#include <stdlib.h>
#include <conio.h>
using namespace std;
struct node
{
    int data;
    struct node *next;
};
node *getNode(int data)
{
    /*This function creates a node and enter a data into it*/
    node *newNode = new node();
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
struct node *START = NULL;
node *Ins_beg(node *START, int val)
{
    /*This function inserts a node into beginning */
    struct node *newNode = new node();
    newNode->data = val;
    newNode->next = START;
    START = newNode;
    return START; // Returns the START pointer
}
node *Ins_End(node *START, int val)
{
    struct node *newNode = new struct node;
    newNode->data = val;
    newNode->next = NULL;
    if (START == NULL)
        START = newNode;
    else
    {
        struct node *temp = START;
        while (temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    return START;
}
```

```
node *InsAt_Given_Loc(node *START, int info, int loc)
{
    struct node *temp = START;
    struct node *newNode = new node();
    newNode->data = info;
    if (loc == 1)
    {
        newNode->next = START;
        START = newNode;
    }
    else if (loc >= 2 && temp != NULL)
    {
        for (int i = 1; i <= loc - 2; i++)
        {
            temp = temp->next;
            if (temp == NULL)
                cout << "\nLocation is greater\n";
        }
        newNode->next = temp->next;
        temp->next = newNode;
    }
    else
        cout << "\nInvalid Location\n";
    return START;
}
void Print_List(struct node *START)
{
    struct node *temp = START;
    while (temp != NULL)
    {
        cout << temp->data << " ";
        temp = temp->next;
    }
}
int main()
{
    int ch, value,loc;
    node *START = NULL, *t1 = START,*t2=START,*t3=START;
    // struct node *START;
    START = getNode(10);                    // First node
    START->next = getNode(20);              // Second node
    START->next->next = getNode(30);        // Third node
    START->next->next->next = getNode(40); // Fourth node
xx:
    cout << "\n\t|>|\t HELLO FRIENDS THIS IS 201B172";
    cout << "\n\n\t\t|>|\t\t This Is Your Basic List : ";
```

```
    Print_List(START); // Print the basic List
    cout << "\n\n\n\t\t|>|\t\tPress 1 : To Insert Node at the
Beginning :";
    cout << "\n\t\t|>|\t\tPress 2 : To Insert Node at the End :";
    cout << "\n\t\t|>|\t\tPress 3 : To Insert Node at the
Spec_Location :";
    cout << "\n\t\t|>|\t\tPress 4 : To Exit :-) :";
    cout << "\n\n\n\n\t\t|>|\t\tKindly Enter your Choice : ";
    cin >> ch;
    switch (ch)
    {
    case 1:
        system("cls");
        cout << "\n\t|>|\t HELLO FRIENDS THIS IS 201B172";
        cout << "\n\n\n\t\t|>|\t\tYou Are Inserting a Node at the
Beginning :";
        cout << "\n\n\t\t|>|\t\tKindly Enter your Value  : ";
        cin >> value;
        t1 = Ins_beg(START, value);
        cout << "\n\n\t\t|>|\t\t Your List Output : ";
        Print_List(t1);
        cout << "\n\n\t\t|>|\t\t PRESS ANY KEY TO RETURN : ";
        getch();
        system("cls");
        goto xx;
        break;
    case 2:
        system("cls");
        cout << "\n\t|>|\t HELLO FRIENDS THIS IS 201B172";
        cout << "\n\n\n\t\t|>|\t\tYou Are Inserting a Node at the
End :";
        cout << "\n\n\t\t|>|\t\tKindly Enter your Value  : ";
        cin >> value;
        t2 =Ins_End(START,value);
        cout << "\n\n\t\t|>|\t\t Your List Output : ";
        Print_List(t2);
        // delete t2;
        cout << "\n\n\t\t|>|\t\t PRESS ANY KEY TO RETURN : ";
        getch();
        system("cls");
        goto xx;
        break;
    case 3:
        system("cls");
        cout << "\n\t|>|\t HELLO FRIENDS THIS IS 201B172";
```

```
        cout << "\n\n\n\t\t|>|\t\tYou Are Inserting a Node at the
Spec_Loaction :";
        cout << "\n\n\t\t|>|\t\tKindly Enter your Value  : ";
        cin >> value;
        cout << "\n\t\t|>|\t\tKindly Enter Location  : ";
        cin>>loc;
        t3 =InsAt_Given_Loc(START,value,loc);
        cout << "\n\n\t\t|>|\t\t Your List Output : ";
        Print_List(t3);
        cout << "\n\n\t\t|>|\t\t PRESS ANY KEY TO RETURN : ";
        getch();
        system("cls");
        goto xx;
        break;
    case 4:
        exit(0);
    default:
        cout<<"\n *-* Your Response is Invalid *-* \n";
        break;
    }

    return 0;
}
```

```
/*Write a menu driven program using switch-case to Delete the node
at beginning,
after specified position and at the end of linked list.*/
#include <iostream>
#include <stdlib.h>
#include <conio.h>
using namespace std;
struct node
{
    int data;
    struct node *next;
};
node *getNode(int data)
{
    /*This function creates a node and enter a data into it*/
    node *newNode = new node();
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
```

```
}
struct node *START = NULL;
node *Del_beg(node *START)
{
    struct node *temp;
    if (START == NULL)
    {
        cout << "\nEMPTY";
        return START; // Returns the START pointer
    }
    else
    {
        temp = START;
        START = START->next;
        return START; // Returns the START pointer
        delete (temp);
    }
}
node *Del_End(node *START)
{
    node *temp, *temp1;
    temp1 = START;
    // temp = START;
    if (START == NULL)
    {
        cout << "\nEMPTY";
        return NULL;
    }
    else
    {
        while (temp1->next != NULL)
        {
            temp = temp1;
            temp1 = temp1->next;
        }
        temp->next = NULL;
        delete (temp1);
        return START;
    }
}
node *DelAft_Given_Loc(node *START, int loc)
{
    node *temp = START, *temp1 = START;
    if (START == NULL || loc <= 0)
    {
        cout << "\nEMPTY";
```

```
        return START;
    }
    if (loc == 1)
    {
        START = START->next;
        delete (temp1);
    }
    else if (loc >= 2 && START->next != NULL)
    {
        for (int i = 1; i <= loc - 1; i++)
        {
            temp = temp->next;
            if (temp->next == NULL)
            {
                cout << "\n\n\t\t|>|\t\t||You are trying to to
delete from invalid position :-(  ||Loc is greater|| : ";
                // cout << "\n\n\t\t|>|\t\t Loc is greater : ";
                // cout << "\nLoc is greater ";
                return START;
            }
        }
        temp1 = temp->next;
        temp->next = temp1->next;
        delete (temp1);
        return START;
    }
    else
        cout << "\n\n\t\t|>|\t\t||You are trying to to delete from
invalid position :-(  ||Loc is greater|| : ";
    // cout << "\nLoc is greater ";
}
void Print_List(struct node *START)
{
    struct node *temp = START;
    while (temp != NULL)
    {
        cout << temp->data << " ";
        temp = temp->next;
    }
}
int main()
{
    int ch, loc;
    node *START = NULL, *t1 = START, *t2 = START, *t3 = START;
    // struct node *START;
    START = getNode(10);                    // First node
```

```
    START->next = getNode(20);             // Second node
    START->next->next = getNode(30);       // Third node
    START->next->next->next = getNode(40); // Fourth node
xx:
    cout << "\n\t|>|\t HELLO FRIENDS THIS IS 201B172";
    cout << "\n\n\t\t|>|\t\t This Is Your Basic List : ";
    Print_List(START); // Print the basic List
    cout << "\n\n\n\t\t|>|\t\tPress 1 : To Delete Node from the
Beginning :";
    cout << "\n\t\t|>|\t\tPress 2 : To Delete Node from the End :";
    cout << "\n\t\t|>|\t\tPress 3 : To Delete Node After the
Spec_Location :";
    cout << "\n\t\t|>|\t\tPress 4 : To Exit :-) :";
    cout << "\n\n\n\n\t\t|>|\t\tKindly Enter your Choice : ";
    cin >> ch;
    switch (ch)
    {
    case 1:
        system("cls");
        cout << "\n\t|>|\t HELLO FRIENDS THIS IS 201B172";
        cout << "\n\n\n\t\t|>|\t\tYou Are Deleting a Node from the
Beginning :";
        t1 = Del_beg(START);
        cout << "\n\n\t\t|>|\t\t Your List Output : ";
        Print_List(t1);
        cout << "\n\n\t\t|>|\t\t PRESS ANY KEY TO RETURN : ";
        getch();
        system("cls");
        goto xx;
        break;
    case 2:
        system("cls");
        cout << "\n\t|>|\t HELLO FRIENDS THIS IS 201B172";
        cout << "\n\n\n\t\t|>|\t\tYou Are Deleting a Node from the
End :";
        t2 = Del_End(START);
        cout << "\n\n\t\t|>|\t\t Your List Output : ";
        Print_List(t2);
        cout << "\n\n\t\t|>|\t\t PRESS ANY KEY TO RETURN : ";
        // delete t2;
        getch();
        system("cls");
        goto xx;
        break;
    case 3:
        system("cls");
```

```
        cout << "\n\t|>|\t HELLO FRIENDS THIS IS 201B172";
        cout << "\n\n\n\t\t|>|\t\tYou Are Deleting a Node After the
Spec_Loaction :";
        cout << "\n\t\t|>|\t\tKindly Enter Location  : ";
        cin >> loc;
        t3 = DelAft_Given_Loc(START, loc);
        cout << "\n\n\t\t|>|\t\t Your List Output : ";
        Print_List(t3);
        cout << "\n\n\t\t|>|\t\t PRESS ANY KEY TO RETURN : ";
        getch();
        system("cls");
        goto xx;
        break;
    case 4:
        exit(0);
    default:
        cout << "\n *-* Your Response is Invalid *-* \n";
        break;
    }

    return 0;
}
```

```
// WAP to reverse the singly linked list.
#include <bits/stdc++.h>
using namespace std;

// A linked list Node
struct node
{
    int data;
    struct node *next;
};

// function to create and return a Node
node *getNode(int data)
{
    // allocating space
    node *newNode = new node();

    // inserting the required data
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
```

```
node *Reverse(node *START)
{
    node *Cnode, *Pnode, *Nnode;
    if (START == NULL)
        return START;
    Cnode = START;
    Nnode = Cnode->next;
    Cnode->next = NULL;
    while (Nnode != NULL)
    {
        Pnode = Cnode;
        Cnode = Nnode;
        Nnode = Nnode->next;
        Cnode->next = Pnode;
    }
    START = Cnode;
    return START;
}
// This function prints contents
// of the linked list
void printList(struct node *START)
{
    while (START != NULL)
    {
        cout << " " << START->data;
        START = START->next;
    }
    cout << endl;
}

// Driver Code
int main()
{
    // Creating the list 3->5->8->10
    node *START,*t1;
    START = getNode(101);
    START->next = getNode(202);
    START->next->next = getNode(303);
    START->next->next->next = getNode(404);
    cout<<"\nHere is Your Linked List : ";
    printList(START);
    t1=Reverse(START);
    cout<<"\nHere is Your Reversed Linked List : ";
    printList(t1);

    return 0;
```

```
}
```

```
/*WAP to search an element in the linked list.*/
#include <iostream>
using namespace std;
struct node
{
    int data;
    struct node *next;
};
node *getNode(int data)
{
    /*This function creates a node and enter a data into it*/
    node *newNode = new node();
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
int Search(node *START, int info)
{
    node *temp;
    int loc = 0;
    temp = START;
    if (temp == NULL)
        return -1;
    while (temp != NULL)
    {
        loc++;
        if (temp->data == info)
            return loc;
        temp = temp->next;
    }
    return -1;
}
void Print_List(node *START)
{
    struct node *temp = START;
    while (temp != NULL)
    {
        cout << temp->data << " ";
        temp = temp->next;
    }
}
int main()
{
    node *START = NULL;
```

```
    int info;                                  // *t2 = START;
    START = getNode(10);                       // First node
    START->next = getNode(20);                 // Second node
    START->next->next = getNode(30);           // Third node
    START->next->next->next = getNode(40); // Fourth node
    Print_List(START);
    cout << "\nEnter the element you want to search : \n";
    cin >> info;
    if (Search(START, info) == -1)
        cout << "Error" << endl;
    else
        cout << "Element is at : " << Search(START, info);
    return 0;
}
```

```
// Assume you have given a start pointer of a singly linked list.
Write a program
//  to find the middle node in the given linked list.
#include <iostream>
using namespace std;
struct node
{
    int data;
    struct node *next;
};
node *getNode(int data)
{
    /*This function creates a node and enter a data into it*/
    node *newNode = new node();
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
void printMiddle(struct node *head)
{
    struct node *slow_ptr = head;
    struct node *fast_ptr = head;

    if (head != NULL)
    {
        while (fast_ptr != NULL && fast_ptr->next != NULL)
        {
            fast_ptr = fast_ptr->next->next;
            slow_ptr = slow_ptr->next;
        }
```

```
        cout << "\nThe middle element is [" << slow_ptr->data <<
"]" << endl;
    }
}
void Print_List(node *START)
{
    struct node *temp = START;
    while (temp != NULL)
    {
        cout << temp->data << " ";
        temp = temp->next;
    }
}
int main()
{
    node *START = NULL;
    int info;
    START = getNode(10);                          // First node
    START->next = getNode(20);                    // Second node
    START->next->next = getNode(30);              // Third node
    START->next->next->next = getNode(40);        // Fourth node
    START->next->next->next->next = getNode(50); // Fifth node
    cout << "\nYour List : \n";
    Print_List(START);
    printMiddle(START);
    return 0;
}
```