Marwadi University	**Marwadi University** **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: DSC (01CT0308)	**Aim:** Implementations of Linked Lists menu-driven program. operations on the linked list – copy, concatenate, split, reverse, count no. of nodes, etc.	
Experiment No: 1	**Date:** **25 - 08 - 2023**	**Enrolment No:-** 92200133030

Experiment – 1

Objective: Implementations of Linked Lists menu-driven program. operations on the linked list – copy, concatenate, split, reverse, count no. of nodes, etc.

Code :-

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

      node(int val) {
         data = val ;
         next = NULL ;
      }};
class linkedlist {
   public:
      node* head;
      linkedlist() {
         head = NULL ;}
      void insertAtHead(int val, node* &head) {
         node* n = new node(val);
         n->next = head;
         head = n;}
      void insertAtTail(int val, node* &head) {
         node* n = new node(val);
         if (head == NULL) {
            node* n = new node(val);
            n->next = head;
            head = n;
            return;}
```

```
        node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;}
        temp->next = n;}
    void insertInBetweenData(node* &head, int datain, int val) {
        node* temp = head;
        while (temp != NULL && temp->data != datain) {
            temp = temp->next;}
        if (temp == NULL) {
            cout << "The Node With Data " << datain << " Is Not Available :- " << endl;
            return;}
        node* n = new node(val);
        n->next = temp->next;
        temp->next = n;
    }
    void insertInBetweenIndex(node* &head, int index, int val) {
        if (index == 0) {
            insertAtHead(val, head);
            return;}
        node* temp = head;
        int tempindex = 0;
        while (temp != NULL && tempindex < index - 1) {
            temp = temp->next;
            tempindex++;}
        if (temp == NULL) {
            cout << "Index Out Of The Bound." << endl;
            return;}
        node* n = new node(val);
        n->next = temp->next;
        temp->next = n;}
    void display(node* head) {
        while (head != NULL) {
            cout << head->data << " -> ";
            head = head->next;}
        cout << " NULL " << endl;}
    bool checkbydata(node* head, int datain, int length) {
        for (int i = 0; i < length; i++) {
            if (head->data == datain) {
                return true;}
            head = head->next;}
        return false;}
    int lengthofLL(node* head) {
        int length = 0;  // Start the length from 0
        node* temp = head;
while (temp != NULL) {
            length++;
            temp = temp->next;}
```

```
    return length;}
void deletehead(node* &head) {
    node* temp = head;
    head= head->next ;
    temp->next = NULL;
    delete temp ;
    return ;}
void deletebydata(node* &head,int datain) {
    if(head == NULL) {
        cout << "LinkedList Is Empty." << endl ;}
    if(head->next == NULL) {
        deletehead(head) ;}
    node* temp = head ;
    while(temp->next->data != datain) {
        if(temp->next == NULL) {
            cout << "Not Found The Node With Data " << datain << endl ;}
        temp = temp->next ;}
    node* todelete = temp->next ;
    temp->next = todelete->next ;
    delete todelete ; }
void deletebyindex(node* &head,int index) {
    if(index < lengthofLL(head)) {
        int tempindex = 0 ;
        node* temp = head ;
        while(tempindex < index - 1) {
            temp = temp->next ;
            tempindex++;}
        node* todelete = temp->next ;
        temp->next = todelete->next ;
        delete todelete ;
        return ;}
    else {
        cout << index << " Is Not Exist In LinkedList." << endl ;
        return ;}}
bool checkbyindex(int length, int index) {
    return index < length;}
void reverseLL(node* &head) {
    node* current = head ;
    node* prev = NULL ;
    node* next = NULL ;
        while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;}
    head = prev;}
node* reverseknodes(node* &head,int k) {
```

```
        node* prevptr = NULL ;
        node* currptr = head;
node* nextptr;
        int count = 0 ;
        while(currptr!= NULL && count < k) {
           nextptr = currptr->next ;
           currptr->next = prevptr ;
           prevptr = currptr ;
           currptr = nextptr ;
           count++;}
        if(nextptr != NULL) {
           head->next = reverseknodes(nextptr,k);}
        return prevptr ;}
      bool detectcycle(node* &head) {
         node* fast = head ;
         node* slow = head;
        while(fast->next != NULL && fast != NULL) {
           fast = fast->next->next ;
           slow = slow->next ;
           if(fast == slow) {
              cout << "There Exist A Cycle In Youe LinkedList." << endl ;
              return true ;}}
        cout << "No Cycle Exist." <<endl;
        return false ;}
      void removecycle(node* &head) {
        if(!detectcycle(head)) {
           node* fast = head;
           node* slow = head ;
           do{
              slow = slow->next ;
              fast = fast->next->next ;
           } while (fast != slow);
           fast = head ;
           while(slow->next != fast->next) {
              slow = slow->next ;
              fast = fast->next ;}
           slow->next = NULL ;}
        else {
           cout << "No Cycle Exist." << endl ;}}};
int main() {
   int choice ;
   linkedlist l ;
   while(choice != 16) {
      cout << "Enter The Choice As Per The List Given Below :-\n1)Insert The Element At
Head\n2)Insert The Element At Tail\n3)Insert After Data\n4)Insert On Index\n5)Display The
List\n6)Check Node With Data Is Present\n7)Check Index Is Present Or Not\n8)Find The
Length Of LinkedList\n9)Delete Head\n10)Delete Node With Given Data\n11)Delete Node
```

```
With Given Index\n12)Reverse The Linked List.\n13)Reverse K Nodes\n14)Detect Cycle In
The LinkedList.\n15)Remove Cycle In The LinkedList.\n16)Exit." <<endl ;
        cout << "Enter Your Choice :- " ;
        cin >> choice ;
        switch (choice){
            case 1 : {
                int val ;
                cout << "Enter The Data To Insert At Head Of Your LinkedList :- " ;
                cin >> val ;
                l.insertAtHead(val , l.head);
                break;}
            case 2 : {
                int val ;
                cout << "Enter The Data To Insert At Tail Of Your LinkedList :- " ;
                cin >> val ;
                l.insertAtTail(val , l.head);
                break;}
            case 3 : {
                int val , ref ;
                cout << "Enter The Data of Node After Which You Want To Insert A Node :- " ;
                cin >> ref ;
                cout << "Enter The Data of Your New Node :- " ;
                cin >> val;
                l.insertInBetweenData(l.head,ref,val);
                break;}
            case 4 : {
                int val ;
                int index;
                cout << "Enter The Index After Which You Want to Insert A Node :- "  ;
                cin >> index ;
                cout << "Enter The Data Of Your New Node :- " ;
                cin >> val ;
                l.insertInBetweenIndex(l.head,index,val);
                break;}
            case 5 : {
                cout << "Your Linked List Is :- " << endl;
                l.display(l.head);
                break;}
            case 6 : {
                int val ;
                cout << "Enter The Data of Node Which Presence You Want To Check :- " ;
                cin >> val ;
                l.checkbydata(l.head , val , l.lengthofLL(l.head));
                break;}
            case 7 : {
                int index ;
                cout << "Enter The Index Which Presence You Want To Check :- " ;
```

```
            cin >> index ;
            l.checkbyindex(l.lengthofLL(l.head) , index);
            break;}
        case 8 : {
            int val ;
            cout << "The Length of Your LinkedList Is :- " ;
            l.lengthofLL(l.head);
            break;}
        case 9 : {
            cout << "Your Head Of Your LinkedList Is Deleted." ;
            l.deletehead(l.head);
            break;}
        case 10 : {
            int val ;
            cout << "Enter The Data of Node Which You Want to Delete :- " ;
            cin >> val ;
            l.deletebydata(l.head,val);
            break;}
        case 11 : {
            int index ;
            cout << "Enter The Index of Node Which You Want To Delete From Your
LinkedList :- " ;
            cin >> index ;
            l.deletebyindex(l.head,index);
            break ; }
        case 12 : {
            cout << "Your Linkedlist Is Reversed." << endl ;
            l.reverseLL(l.head);
            break;}
        case 13 : {
            int k;
            cout << "Enter The Value Of K :-" << endl ;
            cin >> k ;
            l.reverseknodes(l.head ,k);
            break;}
        case 14 : {
            l.detectcycle(l.head);
            break;}
        case 15 : {
            l.removecycle(l.head);
            cout << "Cycle Is Removed From Your Linked List." << endl ;
            break;}
        case 16 : {
            cout << "Programm Ends." << endl ;
            break;}
        default: {
            cout << "Enter A Valid Choice." <<endl;
```

```
        break;}}
    if(choice == 13) { break; }}}
```

Output:

```
Enter The Choice As Per The List Given Below :-
1)Insert The Element At Head
2)Insert The Element At Tail
3)Insert After Data
4)Insert On Index
5)Display The List
6)Check Node With Data Is Present
7)Check Index Is Present Or Not
8)Find The Length Of LinkedList
9)Delete Head
10)Delete Node With Given Data
11)Delete Node With Given Index
12)Reverse The Linked List.
13)Reverse K Nodes
14)Detect Cycle In The LinkedList.
15)Remove Cycle In The LinkedList.
16)Exit.
Enter Your Choice :- 1
Enter The Data To Insert At Head Of Your LinkedList :- 123
Enter The Choice As Per The List Given Below :-
1)Insert The Element At Head
2)Insert The Element At Tail
3)Insert After Data
4)Insert On Index
5)Display The List
6)Check Node With Data Is Present
7)Check Index Is Present Or Not
8)Find The Length Of LinkedList
9)Delete Head
10)Delete Node With Given Data
11)Delete Node With Given Index
12)Reverse The Linked List.
13)Reverse K Nodes
14)Detect Cycle In The LinkedList.
15)Remove Cycle In The LinkedList.
16)Exit.
Enter Your Choice :- 2
Enter The Data To Insert At Tail Of Your LinkedList :- 234
```

```
Enter Your Choice :- 2
Enter The Data To Insert At Tail Of Your LinkedList :- 234
Enter The Choice As Per The List Given Below :-
1)Insert The Element At Head
2)Insert The Element At Tail
3)Insert After Data
4)Insert On Index
5)Display The List
6)Check Node With Data Is Present
7)Check Index Is Present Or Not
8)Find The Length Of LinkedList
9)Delete Head
10)Delete Node With Given Data
11)Delete Node With Given Index
12)Reverse The Linked List.
13)Reverse K Nodes
14)Detect Cycle In The LinkedList.
15)Remove Cycle In The LinkedList.
16)Exit.
Enter Your Choice :- 1
Enter The Data To Insert At Head Of Your LinkedList :- 145
Enter The Choice As Per The List Given Below :-
1)Insert The Element At Head
2)Insert The Element At Tail
3)Insert After Data
4)Insert On Index
5)Display The List
6)Check Node With Data Is Present
7)Check Index Is Present Or Not
8)Find The Length Of LinkedList
9)Delete Head
10)Delete Node With Given Data
11)Delete Node With Given Index
12)Reverse The Linked List.
13)Reverse K Nodes
14)Detect Cycle In The LinkedList.
15)Remove Cycle In The LinkedList.
16)Exit.
Enter Your Choice :- 5
Your Linked List Is :-
145 -> 123 -> 234 ->  NULL
```