

**Symbiosis Institute of Technology**

**Faculty of Engineering**

**CSE- Academic Year 2024-25**

**Data Structures – Lab Batch 2023-27**

|  |  |
| --- | --- |
| **Lab Assignment No:- 6** | |
|  | |
| **Name of Student** | Pankhuri Varshney |
| **PRN No.** | 23070122160 |
| **Batch** | 2023-27 |
| **Class** | CS-B2 |
| **Academic Year & Semester** | 2024-25  Semester 3 |
| **Date of Performance** | 1st October, 2024 |
|  | |
| **Title of Assignment:** | Write a program for:   1. Creation of Doubly Linked list 2. Insertion at beginning 3. Insertion at end 4. Insertion after specific node 5. Display |
| **Source Code/Algorithm/Flow Chart:** | **Implement a program to:**   1. **Creation of Doubly Linked list** 2. **Insertion at beginning** 3. **Insertion at end** 4. **Insertion after specific node** 5. **Display**   **SOURCE CODE:**  #include <iostream>  using namespace std;  class Node{  public:  int data;  Node \*left;  Node \*right;  Node(int data){  this->data=data;  this->left=this->right=NULL;  }  };  class DLL{  private:  Node \*head;  Node \*tail;  public:  DLL(){  this->head=NULL;  this->tail=NULL;  }  public:  void addNodeEnd(int data){  Node \*newNode=new Node(data);  if(tail==NULL){  tail=head=newNode;  return;  }  tail->right=newNode;  newNode->left=tail;  tail=newNode;  }  void addNodeBegin(int data){  Node \*newNode=new Node(data);  if(head==NULL){  head=tail=newNode;  return;  }  newNode->right=head;  head->left=newNode;  head=newNode;  }  void addNodeAt(int data, int pos){  Node \*newNode=new Node(data);  if(pos==1){  addNodeBegin(data);  return;  }  int i=1;  Node \*temp=head;  while(temp!=NULL&&pos!=i){  temp=temp->right;  i++;  }  if(temp==NULL){  cout<<"POSITION OUT OF BOUNDS\n";  return;  }  newNode->right=temp;  newNode->left=temp->left;  temp->left->right=newNode;  temp->left=newNode;  }  void deleteNodeEnd(){  if(tail==NULL){  return;  }  Node \*del=tail;  tail=tail->left;  tail->right=NULL;  delete(del);  }  void deleteNodeBegin(){  if(head==NULL){  return;  }  Node \*del=head;  head=head->right;  head->left=NULL;  delete(del);  }  void deleteNodeAt(int pos){  if(head==NULL){  return;  }  if(pos==1){  deleteNodeBegin();  return;  }  int i=1;  Node \*temp=head;  while(temp!=NULL&&pos!=i){  temp=temp->right;  i++;  }  if(temp==NULL){  cout<<"POSITION OUT OF BOUNDS\n";  return;  }  temp->left->right=temp->right;  temp->right->left=temp->left;  delete(temp);  }  void display(){  if(head==NULL||tail==NULL){  cout<<"EMPTY LIST"<<endl;  return;  }  cout<<"From left to right: \nNULL <-> ";  Node \*temp=head;  while(temp!=NULL){  cout<<temp->data<<" <-> ";  temp=temp->right;  }  cout<<"NULL"<<endl;  cout<<"From right to left: \nNULL <-> ";  temp=tail;  while(temp!=NULL){  cout<<temp->data<<" <-> ";  temp=temp->left;  }  cout<<"NULL\n"<<endl;  }  };  int main(){  cout<<"DOUBLE LINKED LIST:"<<endl;  DLL \*dll=new DLL();  dll->addNodeEnd(20);  dll->addNodeEnd(40);  dll->addNodeEnd(30);  dll->addNodeBegin(10);  dll->addNodeBegin(50);  dll->addNodeBegin(60);  cout<<"AFTER CREATION: "<<endl;  dll->display();  dll->addNodeAt(80,3);  dll->addNodeAt(90,5);  cout<<"AFTER INSERTION AT SPECIFIC INDEX: "<<endl;  dll->display();  dll->deleteNodeEnd();  dll->deleteNodeBegin();  cout<<"AFTER DELETION:"<<endl;  dll->display();  dll->deleteNodeAt(1);  dll->deleteNodeAt(3);  cout<<"AFTER DELETION AT SPECIFIC INDEX:"<<endl;  dll->display();  } |
| **Output Screenshots** |  |
| **Practice questions** |  |
| **Conclusion** | Thus, we have studied the concept of Linked List and how it is different from arrays. |