**Instructions: Please read carefully**

* Please rename this file as only your ID number **(e.g. 18-\*\*\*\*\*-1.doc or 18-\*\*\*\*\*-1.pdf).**
* Submit the file before **11:59pm on 28/07/2020** in the Portal Lab Performance section labeled **Lab task 9. If you cannot complete the full task, do not worry. Just upload what you have completed.**

|  |
| --- |
| **1. Do the following to write program for a Single Linked List:**     1. Create a singly linked list by **inserting** node one by one at the end. 2. **Display** your list 3. **Search** an item into your linked list. 4. **Insert** a new item at a specific position (at the beginning and after a given node) 5. **Delete** an item from the list (at beginning, at last and at middle)   Take users input from the console to perform the operations. |
| **Your code here:**  #include <iostream>  using namespace std;  class Node  {  public:  int data ;  Node\* next ;  };  class List:public Node{  Node \*first,\*last ;  public:  List()  {  first = NULL ;  last = NULL ;  }  void create();  void insert();  void delate();  void display();  void search();  };  void List::create()  {  Node \*temp;  temp=new Node;  int n;  cout<<"\nEnter an Element:";  cin>>n;  temp->data=n;  temp->next=NULL;  if(first==NULL)  {  first=temp;  last=first;  }  else  {  last->next=temp;  last=temp;  }  }  void List::insert()  {  Node \*prev,\*cur;  prev=NULL;  cur=first;  int count=1,pos,ch,n;  Node \*temp=new Node;  cout<<"\nEnter an Element:";  cin>>n;  temp->data=n;  temp->next=NULL;  cout<<"\nINSERT AS\n1:FIRSTNODE\n2:LASTNODE\n3:IN BETWEEN FIRST&LAST NODES";  cout<<"\nEnter Your Choice:";  cin>>ch;  switch(ch)  {  case 1:  temp->next=first;  first=temp;  break;  case 2:  last->next=temp;  last=temp;  break;  case 3:  cout<<"\nEnter the Position to Insert:";  cin>>pos;  while(count!=pos)  {  prev=cur;  cur=cur->next;  count++;  }  if(count==pos)  {  prev->next=temp;  temp->next=cur;  }  else  cout<<"\n Not Insert !!!";  break;  }  }  void List::delate()  {  Node \*prev=NULL,\*cur=first;  int count=1,pos,ch;  cout<<"\nDELETE\n1:FIRSTNODE\n2:LASTNODE\n3:IN BETWEEN FIRST&LAST NODES";  cout<<"\nEnter Your Choice:";  cin>>ch;  switch(ch)  {  case 1:  if(first!=NULL)  {  cout<<"\nDeleted Element is "<<first->data;  first=first->next;  }  else  cout<<"\nNot Able to Delete";  break;  case 2:  while(cur!=last)  {  prev=cur;  cur=cur->next;  }  if(cur==last)  {  cout<<"\nDeleted Element is: "<<cur->data;  prev->next=NULL;  last=prev;  }  else  cout<<"\nNot Able to Delete";  break;  case 3:  cout<<"\nEnter the Position of Deletion:";  cin>>pos;  while(count!=pos)  {  prev=cur;  cur=cur->next;  count++;  }  if(count==pos)  {  cout<<"\nDeleted Element is: "<<cur->data;  prev->next=cur->next;  }  else  cout<<"\nNot Delete!!!!";  break;  }  }  void List::display()  {  Node \*temp=first;  if(temp==NULL)  {  cout<<"\n List also Empty";  }  while(temp!=NULL)  {  cout<<temp->data;  cout<<"-->";  temp=temp->next;  }  cout<<"NULL HERE!!!";  }  void List::search()  {  int value,pos=0;  bool flag=false;  if(first==NULL)  {  cout<<"List is Empty";  return;  }  cout<<"Enter the Value to be Searched:";  cin>>value;  Node \*temp;  temp=first;  while(temp!=NULL)  {  pos++;  if(temp->data==value)  {  flag=true;  cout<<"Element"<<value<<"is Found at "<<pos<<" Position";  return;  }  temp=temp->next;  }  if(!flag)  {  cout<<"Element "<<value<<" not Found this List";  }  }  int main()  {  List l;  int c;  while(1)  {  cout<<"Singly link list"<<endl;  cout<<"1. CREATE"<<endl;  cout<<"2. insert"<<endl;  cout<<"3. delete"<<endl;  cout<<"4. search"<<endl;  cout<<"5. Display"<<endl;  cout<<"6. Exit"<<endl;  cout<<"Enter Your Choice:";  cin>>c;  switch(c)  {  case 1:  l.create();  break;  case 2:  l.insert();  break;  case 3:  l.delate();  break;  case 4:  l.search();  break;  case 5:  l.display();  break;  case 6:  return 0;  }  }  return 0;  } |
| **Your whole Screenshot here: (Console Output):** |

|  |
| --- |
| ***2. Solve and submit at least one of the following problems.***   1. *Write a code to implement Doubly Linked List operations* 2. *Implement Stack using Linked List* 3. *Implement Queue using Linked List* 4. *Implement a program to sort the elements in a Linked List* |
| ***Your code here:***  *#include <iostream>*  *using namespace std;*  *struct Node {*  *int data;*  *struct Node\* next;*  *struct Node\* prev;*  *};*  *void insert\_front(struct Node\*\* head, int new\_data)*  *{*  *struct Node\* newNode = new Node;*  *newNode->data = new\_data;*  *newNode->next = (\*head);*  *newNode->prev = NULL;*  *if ((\*head) != NULL)*  *(\*head)->prev = newNode;*  *(\*head) = newNode;*  *}*  *void insert\_After(struct Node\* prev\_node, int new\_data)*  *{*  *if (prev\_node == NULL) {*  *cout<<"Previous node is required , it cannot be NULL";*  *return;*  *}*  *struct Node\* newNode = new Node;*  *newNode->data = new\_data;*  *newNode->next = prev\_node->next;*  *prev\_node->next = newNode;*  *newNode->prev = prev\_node;*  *if (newNode->next != NULL)*  *newNode->next->prev = newNode;*  *}*  *void insert\_end(struct Node\*\* head, int new\_data)*  *{*  *struct Node\* newNode = new Node;*  *struct Node\* last = \*head;*  *newNode->data = new\_data;*  *newNode->next = NULL;*  *if (\*head == NULL) {*  *newNode->prev = NULL;*  *\*head = newNode;*  *return;*  *}*  *while (last->next != NULL)*  *last = last->next;*  *last->next = newNode;*  *newNode->prev = last;*  *return;*  *}*  *void displayList(struct Node\* node) {*  *struct Node\* last;*  *while (node != NULL) {*  *cout<<node->data<<"-->";*  *last = node;*  *node = node->next;*  *}*  *if(node == NULL)*  *cout<<"NULL";*  *}*  *int main() {*  *struct Node\* head = NULL;*  *insert\_end(&head, 30);*  *insert\_front(&head, 25);*  *insert\_front(&head, 20);*  *insert\_front(&head, 10);*  *insert\_front(&head, 5);*  *insert\_end(&head, 35);*  *insert\_After(head->next, 15);*  *cout<<"Doubly Linked List "<<endl;*  *displayList(head);*  *return 0;*  *}* |
| ***Your whole Screenshot here: (Console Output):*** |