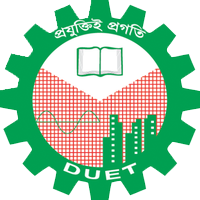
**DHAKA UNIVERSITY OF ENGINEERING & TECHNOLOGY, GAZIPUR**



Department of Computer Science and Engineering

Course No.: **CSE-2112**

Course Title: **Object Oriented Programming Language Sessional**

*Lab No:* **06**

*Lab Name:* Linked List using pointer.

Experiment Date: **27-04-2021** Submission Date: **03-05-2021**

***Submitted By-***

Name: **Mehedi Hasan Shuvo**

Student Id.: 194016

Year: 2nd

Semester: 1st

Session: 2019 - 2020

***Submitted To:***

**Mr.Md. Omor Farque**

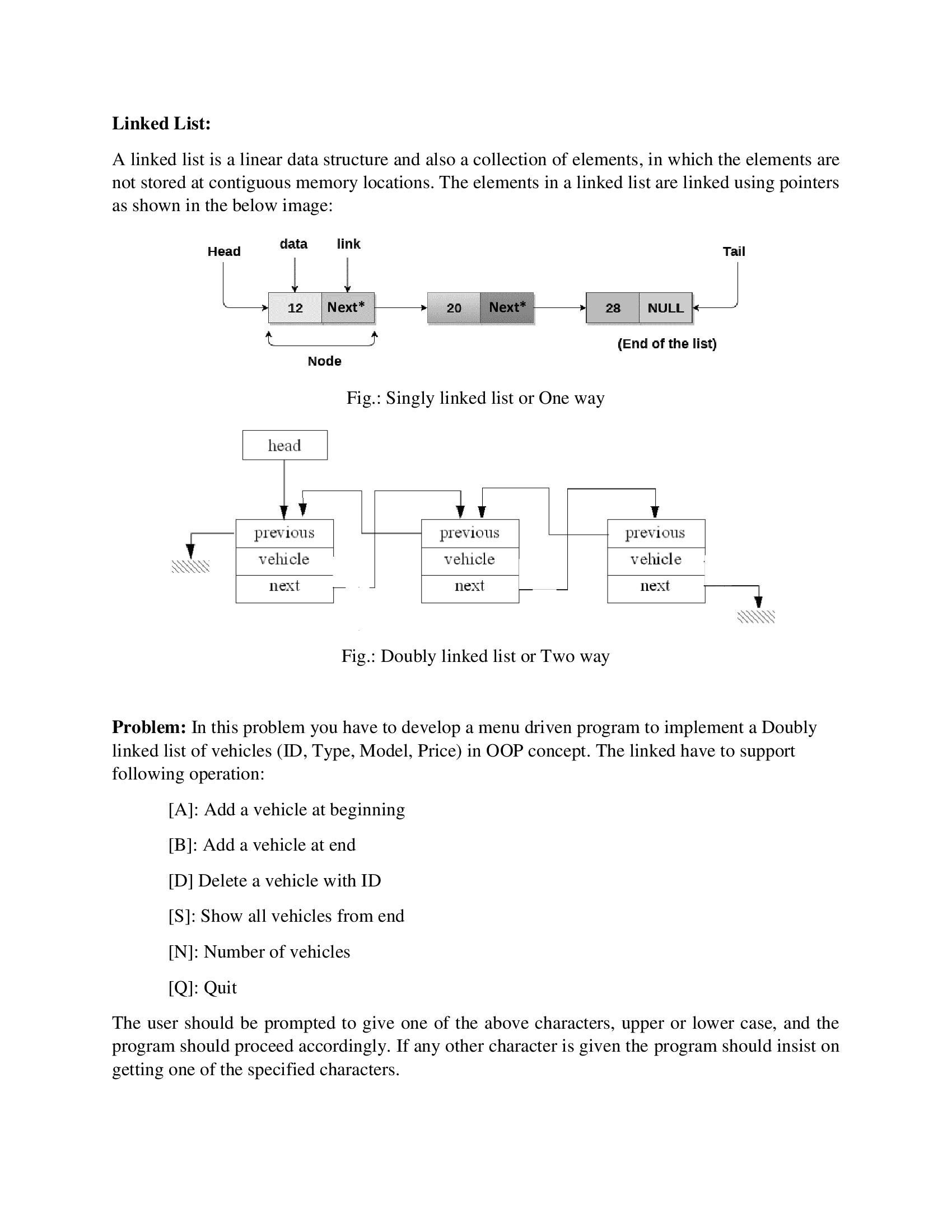
**Associative Professor, Department of CSE**

**Mss. Sabah Binte Noor**

**Associative Professor, Department of CSE**

**Problems No:** 01

**Problem Title:**

****

**Solution:**

|  |
| --- |
| #include<bits/stdc++.h>  using namespace std;  class Vehical  {  public:      int id;      string model;      double price;      string type;      Vehical\* previous;      Vehical\* next;  };  void push(Vehical \*\* head\_ref,int new\_id,string new\_model,string new\_type,double new\_price)  {      Vehical\* new\_vehical=new Vehical();      // put in the data      new\_vehical->id=new\_id;      new\_vehical->model=new\_model;      new\_vehical->price=new\_price;      new\_vehical->type=new\_type;      // make next of new node as head and previously as NULL      new\_vehical->next=(\*head\_ref);      new\_vehical->previous=NULL;      // Change Prev of head node to new node      if((\*head\_ref) !=NULL)      {          (\*head\_ref)->previous=new\_vehical;      }      // Move the head to point to the new node      (\*head\_ref)=new\_vehical;  }  void append(Vehical \*\* head\_ref,int new\_id,string new\_model,string new\_type,double new\_price)  {      //1. Allocate node      Vehical\* new\_vehical=new Vehical();      Vehical\* last=\*head\_ref;    // put in the data      new\_vehical->id=new\_id;      new\_vehical->model=new\_model;      new\_vehical->price=new\_price;      new\_vehical->type=new\_type;      // this new node is going to be the last node, sp make next of it as NULL      new\_vehical->next=NULL;      // if the linked list is empty, then make the new node as head      if(\*head\_ref==NULL)      {          new\_vehical->previous=NULL;          \*head\_ref=new\_vehical;          return;      }      // else traverse till the last node      while(last->next !=NULL)          last=last->next;      // change the next of last node      last->next=new\_vehical;      // make last node as previous of new node      new\_vehical->previous=last;      return;  }  void addAVehical(Vehical \*\*vehial,bool isAddBeganning,bool isAddEnd)  {      int id;      string model;      double price;      string type;      cout<<"\nEnter ID: ";      cin>>id;      cout<<"Enter Vehical Model: ";      cin>>model;      cout<<"Enter Vehical Type: ";      cin>>type;      cout<<"Enter Vehical Price: ";      cin>>price;      cout<<endl;    if(isAddBeganning==true)          push(vehial,id,model,type,price);      if(isAddEnd==true)          append(vehial,id,model,type,price);  }  void deleteNode(Vehical\*\* head\_ref,Vehical\* del)  {      //base case      if(\*head\_ref==NULL || del==NULL) return;      // if node to be deleted is head node      if(\*head\_ref==del) \*head\_ref=del->next;      // change next only if node to be deleted is not the last node      if(del->next !=NULL) del->next->previous=del->previous;      // change prev only if node to be deleted is not the first node      if(del->previous!=NULL) del->previous->next=del->next;      // finally free the memory occupied by delete      free(del);  }  void deleteAllOccurofX(Vehical\*\* head\_ref,int x)  {      //if list is empty      if((\*head\_ref)==NULL) return;      Vehical\* current=\*head\_ref;      Vehical\* next;      // traverse the list up to the end      while (current !=NULL)      {          // if node found with the value of X          if(current->id==x)          {              //save currents next node in the pointer next              next=current->next;              //delete the node pointer to by current              deleteNode(head\_ref,current);              // update current              current=next;          }          else          {              current=current->next;          }      }      cout<<x<<" is successfully deleted form vehical list";  }  int countVehicalList(Vehical\* vehical)  {      int count=0;      Vehical\* last;      while (vehical != NULL)      {          last = vehical;          vehical = vehical->next;          count++;      }      return count;  }  void printVehicalList(Vehical\* vehical)  {      Vehical\* last=NULL;      while (vehical != NULL)      {          last = vehical;          vehical = vehical->next;      }      cout<<"\nID-> Type-> Model-> Price\n";      while (last != NULL)      {          cout<<last->id<<" "<<last->type<<" "<<last->model<<" "<<last->price<<" \n";          last = last->previous;      }  }    int main()  {      Vehical \*head\_vehial=NULL;      char choose;      int query;      while (choose!='Q' || choose!='q')      {          printf("============================================\n");          printf("DOUBLY LINKED LIST PROGRAM\n");          printf("============================================\n");          printf("[A]: Add a vehicle at beginning\n");          printf("[B]: Add a vehicle at end\n");          printf("[D]: Delete a vehicle with ID\n");          printf("[S]: Show all vehicles from end\n");          printf("[N]: Number of vehicles\n");          printf("[Q]: Quit\n");          printf("--------------------------------------------\n");          printf("Enter your choose : ");          cin>>choose;          switch (choose)          {          case 'a':          case 'A':              addAVehical(&head\_vehial,true,false);              break;          case 'b':          case 'B':              addAVehical(&head\_vehial,false,true);              break;          case 'd':          case 'D':              cout<<"Enter Vehical ID: ";              cin>>query;              deleteAllOccurofX(&head\_vehial,query);              break;          case 's':          case 'S':              printVehicalList(head\_vehial);              break;          case 'n':          case 'N':              cout<<"Length of Vehical list is: "<<countVehicalList(head\_vehial);              break;          case 'q':          case 'Q':              return 0;          default:              cout<<"Error! Invalid choice. Please choose between 0-5";          }          printf("\n\n");      }      return 0;  } |
|  |

**Output:**

