**23p-0685 Raqeeb Task 4**

**Question 1:**

#include<iostream>

using namespace std;

class Nodes

{

public:

Nodes \*Next;

Nodes \*Prev;

int ID;

string Name;

int Quantity;

int Price;

Nodes(int value, string name, int quantity, int price)

{

ID= value;

Name=name;

Quantity=quantity;

Next=NULL;

Prev=NULL;

Price= price;

}

};

class Double\_Linked\_list

{

private:

int length;

Nodes \*head;

public:

Double\_Linked\_list()

{

length=0;

head=NULL;

}

void insertion(int value, string name, int quantity,int price)

{

//Insertion with sorting the ID's

Nodes \*ptr1=head;

Nodes \*ptr2=head;

Nodes \*new\_node= new Nodes(value,name,quantity,price);

bool checker=false;

if(head!=NULL)

{

for(int i=1;i<=length;i++)

{

if(ptr2->ID > new\_node->ID)

{

new\_node->Next=ptr2;

if(ptr2==head)

{

head= new\_node;

ptr2->Prev=new\_node;

}

else

{

ptr1->Next=new\_node;

new\_node->Prev=ptr1;

ptr2->Prev=new\_node;

}

checker=true;

break;

}

ptr1=ptr2;

if(ptr2->Next !=NULL)

ptr2=ptr2->Next;

}

if(!checker)

{

ptr2->Next=new\_node;

new\_node->Prev=ptr2;

}

}

else

{

head=new\_node;

}

length+=1;

}

void remove(int ID)

{

if(length==0)

{

cout<<"\nTheir is no product exist !!!!First add the products\n"<<endl;

return;

}

Nodes \*Curr\_ptr=head;

Nodes \*Slow\_ptr=head;

for(int i=1;i<length;i++)

{

if(Curr\_ptr->ID == ID)

{

if(Curr\_ptr == head)

{

head=Curr\_ptr->Next;

delete Curr\_ptr;

length-=1;

return;

}

else

{

Slow\_ptr->Next= Curr\_ptr->Next;

Curr\_ptr->Next->Prev=Slow\_ptr;

delete Curr\_ptr;

length-=1;

return;

}

}

else

{

Slow\_ptr=Curr\_ptr;

Curr\_ptr=Curr\_ptr->Next;

}

}

if(Curr\_ptr->Next == NULL)

{

cout<<"\nNo such ID exist!!!"<<endl;

return;

}

}

void Doubly\_display()

{

if(length==0)

{

cout<<"Their are not products to show first add the products details!!!!!"<<endl;

return;

}

cout<<"\nAll Products Details:\n";

Nodes \*Curr\_ptr=head;

for (int i=1; i<=length;i++)

{

cout<<"ID:"<<Curr\_ptr->ID<<endl;

cout<<"Name:"<<Curr\_ptr->Name<<endl;

cout<<"Quantity:"<<Curr\_ptr->Quantity<<endl;

cout<<"Price:"<<Curr\_ptr->Price<<endl;

if(Curr\_ptr->Next!=NULL)

Curr\_ptr= Curr\_ptr->Next;

cout<<endl;

}

}

void Update\_price(int ID, int price)

{

if(length==0)

{

cout<<"\nTheir are no products yet !!!!\nAdd the products first!!!\n";

return;

}

Nodes \*curr=head;

for(int i=1;i<=length;i++)

{

if(curr->ID==ID)

{

curr->Price=price;

cout<<"\n\*\*\*\*\*\*\*\*\*\nPrice is updated successfully !!!\n\*\*\*\*\*\*\*\*\*"<<endl;

return;

}

curr=curr->Next;

}

cout<<"Invalid ID"<<endl;

}

void find\_product(int id)

{

if(length==0)

{

cout<<"\nTheir are no products to find !!!!\nFirst add the products!!!\n";

return;

}

Nodes \*curr=head;

for(int i=1;i<=length;i++)

{ if(curr->ID==id)

{

cout<<"\nProduct Details:"<<endl;

cout<<"ID:"<<curr->ID<<endl;

cout<<"Product Name:"<<curr->Name<<endl;

cout<<"Quantity:"<<curr->Quantity<<endl;

cout<<"Price:"<<curr->Price<<endl;

return;

}

curr=curr->Next;

}

{

cout<<"Invalid Id such ID does not exist!!!"<<endl;

}

}

};

int main(void)

{

Double\_Linked\_list DL1;

int user\_input;

int ID;

string name;

int quantity;

int price;

//Infinite loop to give end of program control to user

while(1)

{

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"1)Add New Product\n2)Remove Product\n3)Display Product\n4)Update Price\n5)Find Product\n6)Exit";

cout<<"\nUser input:";

cin>>user\_input;

if(user\_input==1)

{

cout<<"Enter Product ID in between (1-10):";

cin>>ID;

cout<<"Enter the Product Name:";

cin.ignore();

getline(cin,name);

cout<<"Enter Quantity of Product:";

cin>>quantity;

cout<<"Enter the price of the product:";

cin>>price;

DL1.insertion(ID,name,quantity,price);

}

else if(user\_input==2)

{

cout<<"Enter the ID of the Product you want to remove:";

cin>>ID;

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

DL1.remove(ID);

}

else if(user\_input==3)

{

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

DL1.Doubly\_display();

}

else if(user\_input==4)

{

cout<<"Enter the ID:";

cin>>ID;

cout<<"Enter the price you want to Update:";

cin>>price;

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

DL1.Update\_price(ID,price);

}

else if(user\_input==5)

{

cout<<"Enter the ID:";

cin>>ID;

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

DL1.find\_product(ID);

}

else if(user\_input==6)

{

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"Thanks for using our services. Have a nice day!"<<endl;

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

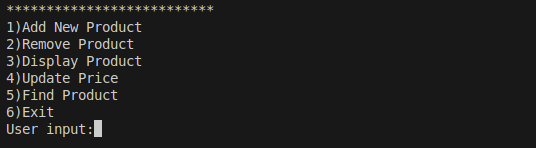
break;

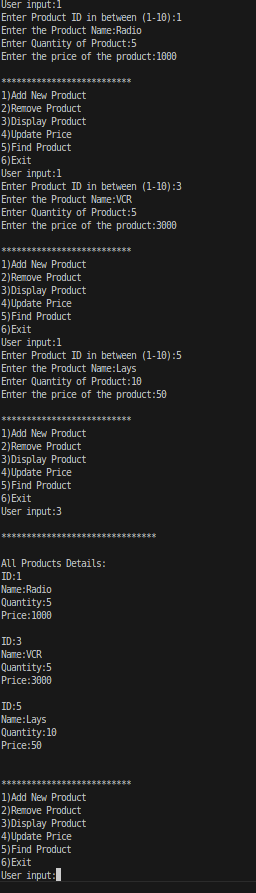
}

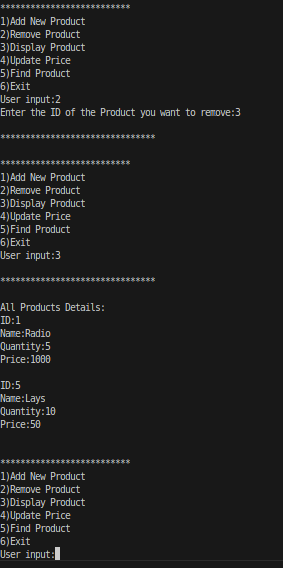
}

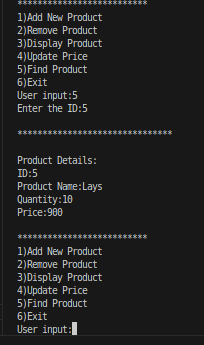
}

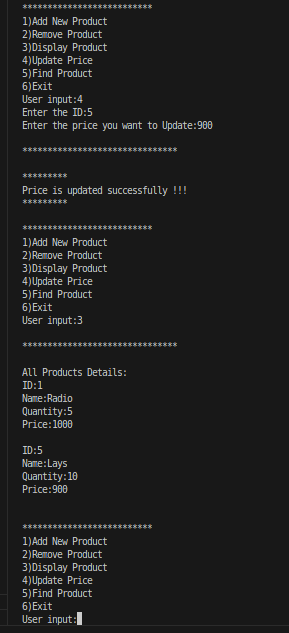
**Output Screen Shots:**

****

****

****

****

****

**Question 2:**

#include<iostream>

using namespace std;

class Nodes

{

public:

Nodes \*next;

int info;

Nodes \*prev;

Nodes(int value)

{

info=value;

next=NULL;

prev=NULL;

}

};

class LinkedList

{

private:

Nodes \*head;

int length;

public:

LinkedList()

{

head=NULL;

length=0;

}

void insertion(int value, int position)

{

//Position check

if(position<0 || position> length+1)

{

cout<<"\nInvalid Position entered!!!!!"<<endl;

return;

}

Nodes \*new\_node= new Nodes(value);

if(position==1)

{

if(head!=NULL)

{

Nodes \*Curr\_ptr=head;

new\_node->next=Curr\_ptr;

Curr\_ptr->prev=new\_node;

head=new\_node;

for(int i=1;i<length;i++)

{

Curr\_ptr=Curr\_ptr->next;

}

}

else

{

head=new\_node;

}

}

else if(position!=length+1)

{

Nodes \*Curr\_ptr=head;

Nodes \*slow2=head;

for (int i=1;i<position;i++)

{

slow2=Curr\_ptr;

Curr\_ptr=Curr\_ptr->next;

}

new\_node->next=Curr\_ptr;

slow2->next=new\_node;

Curr\_ptr->prev=new\_node;

new\_node->prev=slow2;

}

else//For insertion at the end of the circular doubly linkedlist

{ Nodes\*Curr\_ptr=head;

Nodes \*slow2=head;

for (int i=1;i<position-1;i++)

{

Curr\_ptr=Curr\_ptr->next;

}

Curr\_ptr->next=new\_node;

new\_node->prev=Curr\_ptr;

}

length+=1;

}

//To extract the odd values from the original list to the new list

void Odd\_creator(LinkedList OL)

{

Nodes \*Curr1=head;

Nodes \*Curr2=OL.head;

for( int i=1;i<=OL.length;i++)

{

if(Curr2->info % 2 != 0)

{

Nodes \*new\_node= new Nodes(Curr2->info);

if(head==NULL)

{

head=new\_node;

Curr1=new\_node;

}

else

{

Curr1->next=new\_node;

new\_node->prev=Curr1;

Curr1= Curr1->next;

}

length+=1;

}

Curr2=Curr2->next;

}

Curr1->next=head;

}

//To extract the even values from the original list to the new list

void Even\_creator(LinkedList OL)

{

Nodes \*Curr1=head;

Nodes \*Curr2=OL.head;

for( int i=1;i<=OL.length;i++)

{

if(Curr2->info % 2 == 0)

{

Nodes \*new\_node= new Nodes(Curr2->info);

if(head==NULL)

{

head=new\_node;

Curr1=new\_node;

}

else

{

Curr1->next=new\_node;

new\_node->prev=Curr1;

Curr1= Curr1->next;

}

length+=1;

}

Curr2=Curr2->next;

}

Curr1->next=head;

}

void Doubly\_display()

{

Nodes \*Curr\_ptr=head;

for (int i=1; i<=length;i++)

{

cout<<Curr\_ptr->info<<endl;

Curr\_ptr= Curr\_ptr->next;

}

}

};

int main(void)

{

LinkedList OL,L1,L2;

for(int i=1;i<=10;i++)

{

OL.insertion(i,i);

}

OL.Doubly\_display();

L1.Odd\_creator(OL);

L2.Even\_creator(OL);

cout<<"Odd number linked list:"<<endl;

L1.Doubly\_display();

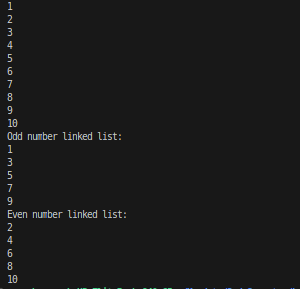
cout<<"Even number linked list:"<<endl;

L2.Doubly\_display();

return 0;

}

**Output Screen Shots:**

****

**Question 3:**

#include<iostream>

using namespace std;

class Nodes

{

public:

Nodes \*Next;

string song\_name;

Nodes(string name)

{

song\_name= name;

Next=NULL;

}

};

class Circular\_singly\_Linked\_list

{

private:

int length;

Nodes \*head;

public:

Circular\_singly\_Linked\_list()

{

length=0;

head=NULL;

}

void insertion(string value, int position)

{

//Position check

if(position<0 || position>length+1)

{

cout<<"INvalid position entered !!!"<<endl;

return;

}

Nodes \*New\_node= new Nodes(value);

if(position==1)

{

if(head!=NULL)

{

New\_node->Next=head;

head=New\_node;

Nodes \*curr\_ptr= head;

for (int i=1;i<=length;i++)

{

curr\_ptr=curr\_ptr->Next;

}

curr\_ptr->Next=New\_node;

}

else

{

head=New\_node;

New\_node->Next=head;

}

}

length+=1;

}

void display\_linked\_list()

{

Nodes \*Curr\_ptr=head;

if(length==0)

{

cout<<"\nPlaylist is empty!!!Enter the songs name first\n";

return;

}

for(int i=1;i<=length;i++)

{

cout<<i<<")"<<Curr\_ptr->song\_name<<endl;

Curr\_ptr=Curr\_ptr->Next;

}

}

//removing song from the end of playlist

string remove()

{

Nodes \*curr1=head;

Nodes \*slow=head;

if(length==0)

{

cout<<"\nThe playlist is empty! First add some songs"<<endl;

return " ";

}

for(int i=0;i<length-1;i++)

{

slow=curr1;

curr1=curr1->Next;

}

slow->Next=head;

string value=curr1->song\_name;

length-=1;

delete curr1;

return value;

}

};

int main(void)

{

Circular\_singly\_Linked\_list CSL1;

int user\_input;

while(1)

{

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*SONGS PLAYLIST\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<"1)Add new song\n2)Remove Song from the end of the list\n3)Display all songs\n4)Quit";

cout<<"\nUser\_input:";

cin>>user\_input;

if(user\_input==1)

{

cout<<"\nEnter the name of song you want to enter:";

string song\_name;

cin.ignore();

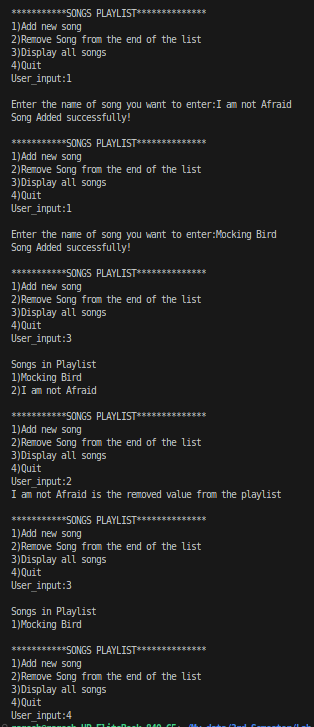
getline(cin,song\_name);

CSL1.insertion(song\_name,1);

cout<<"Song Added successfully!"<<endl;

}

else if(user\_input==2)

{

string check=CSL1.remove();

if(check!=" ")

{

cout<<check<<" is the removed value from the playlist"<<endl;

}

}

else if(user\_input==3)

{

cout<<"\nSongs in Playlist\n";

CSL1.display\_linked\_list();

}

else if(user\_input==4)

{

break;

}

}

}

**Output Screen Shots:**