**Experiment No. 3**

Write a menu-based program that performs following operations:

1. CREATE A LINK LIST
2. INSERTION AT THE BEGINNING
3. INSERTION AT THE END
4. INSERTION AGTER A GIVEN NODE
5. SEARCH A NUMBER IN A LINK LIST
6. FOR CREATE OR INSERTION IN A SORTED LINK LIST
7. DISPLAYING ALL ELEMENTS IN LINK LIST
8. EXIT

**SOURCE CODE**

#include <iostream>

#include<stdlib.h>

using namespace std;

void clean()

{

system("cls");

}

struct node

{

int info;

node \* next;

};

class link

{

private: node \*start;

public: link() //constructor

{

start=NULL;

};

void create();

void insertionb();

void insertione();

void insertiona();

void searching();

void insertions();

void display();

} l; //creating global class variable l

void link::create()

{

clean();

node \* temp=new node;

cout<<"\nENTER THE ELEMENT AT FIRST NODE:";

cin>>temp->info;

temp->next=NULL;

if(start==NULL)

start=temp;

cout<<"\nLINK LIST SUCCESSFULLY CREATED\n\n";

system("pause");

}

void link::insertionb()

{

node \* temp=new node;

cout<<"\nENTER THE ELEMENT:";

cin>>temp->info;

if( start==NULL)

{

temp->next=NULL;

start=temp;

}

else

{

temp->next=start;

start=temp;

}

}

void link::display()

{

cout<<"\n\n";

node \*tmp;

tmp=start;

if(tmp==NULL)

{

cout<<"\nLIST IS EMPTY\n\n";

}

else

{

while(tmp!=NULL)

{

cout<<tmp->info<<" => ";

tmp=tmp->next;

}

cout<<"NULL"<<endl;

}

cout<<"\n\n";

system("pause");

}

void link::insertione()

{

clean();

node \*temp=new node;

node \*temp1=start;

cout<<"\n\nENTER THE ELEMENT:";

cin>>temp->info;

temp->next=NULL;

if(start==NULL)

start=temp;

else

{

while(temp1->next!=NULL)

temp1=temp1->next;

temp1->next=temp;

}

}

void link::insertiona()

{

clean();

int t,a=1;

node \* temp=new node;

node \* temp1=start;

cout<<"LINK LIST \n\n";

l.display();

if(start==NULL)

return;

cout<<"ENTER THE LOCATION WHERE YOU WANT TO ENTER THE ELEMENT:";

cin>>t;

if(t==1)

{

l.insertionb();

}

else

{

cout<<"\nENTER THE ELEMENT YOU WANT TO ENTER:";

cin>>temp->info;

while(a<=t-2)

{

temp1=temp1->next;

a++;

}

temp->next=temp1->next;

temp1->next=temp;}

cout<<"\nLINK LIST AFTER INSERTION\n\n";

l.display();

}

void link::searching()

{

clean();

int item,flag=0,a=1;

node \*tmp=start;

if(tmp==NULL)

{

cout<<"\nLIST IS EMPTY\n\n";

}

else

{

cout<<"\nENTER THE ELEMENT YOU WANT TO SEARCH:";

cin>>item;

while(tmp!=NULL)

{

if(tmp->info==item)

{

flag=a;

}

tmp=tmp->next;

a++;

}

}

if(flag==0)

{

cout<<"\nITEM NOT FOUND\n\n";

}

else

{

cout<<"\nITEM FOUND ON NODE:"<<flag;

cout<<"\n\n";

}

system("pause");

}

void link::insertions()

{

clean();

node \* temp=new node;

node \* ptr=start;

cout<<"ENTER THE ELEMENT:";

cin>>temp->info;

temp->next=NULL;

if(start==NULL)

{

start=temp;

return;

}

else

{

if(start->next==NULL)

{

if(start->info>temp->info)

{

temp->next=start;

start=temp;

}

else

{

start->next=temp;

return;

}

else

{

while(ptr!=NULL)

{

if(ptr->info>=temp->info)

{

temp->next=start;

start=temp;

break;

}

else if(ptr->next==NULL)

{

ptr->next=temp;

return;

}

else if(ptr->info<=temp->info&&ptr->next->info>=temp->info)

{

temp->next=ptr->next;

ptr->next=temp;

return;

}

ptr=ptr->next;

}

}

}

}

int main()

{

int a;

while(1)

{

clean();

cout<<"1.FOR CREATE A LINK LIST\n\n";

cout<<"2.FOR INSERTION AT THE BEGINNING\n\n";

cout<<"3.FOR INSERTION AT THE END\n\n";

cout<<"4.FOR INSERTION AGTER A GIVEN NODE\n\n";

cout<<"5.FOR SEARCH A NUMBER IN A LINK LIST\n\n";

cout<<"6.FOR CREATE OR INSERTION IN A SORTED LINK LIST\n\n";

cout<<"7.FOR DISPLAYING ALL ELEMENTS IN LINK LIST\n\n";

cout<<"8.FOR EXIT\n";

cout<<"\nENTER YOUR CHOISE:";

cin>>a;

switch(a)

{

case 1: l.create();

break;

case 2: l.insertionb();

break;

case 3: l.insertione();

break;

case 4: l.insertiona();

break;

case 5: l.searching();

break;

case 6: l.insertions();

break;

case 7: l.display();

break;

case 8: clean();

cout<<"\nPROGRAM HAS CLOSED SUSSESFULLY\n\n";

exit(0);

default: cout<<"\n\nINVALID OPTION\n\n";

system("pause");

break;

}

}

return 0;

}