Lab #5

Task #1:

#include<iostream>

#include<string>

using namespace std;

class Node

{

public:

float Cgpa;

Node\* Next;

string Name, RollNo;

Node(string name, string roll, float cgpa) //intiazling data through constructor

{

Name = name;

RollNo = roll;

Cgpa = cgpa;

Next = NULL;

}

};

class LinkedList

{

public:

Node \*Head, \*Tail;

LinkedList()

{

Head = NULL;

Tail = NULL;

}

void Insertathead(string name, string roll, float cgpa) //inserting at begning

{

Node \*newnode = new Node(name, roll, cgpa); //creating new node

if (Head == NULL)

{

Head = newnode;

Tail = Head;

}

else

{

newnode->Next = Head;

Head = newnode;

Tail->Next = Head;

}

}

void Insertatlast(string name, string roll, float cgpa) //inserting at last Node

{

Node \*newnode = new Node(name, roll, cgpa);

if (Head == NULL)

{

Head = newnode;

Tail = Head;

}

else

{

Tail->Next = newnode;

Tail = newnode;

Tail->Next = Head;

}

}

void Delete(string roll)

{

Node \*temp = Head;

do

{

if (Head->RollNo == roll)

{

Head = Head->Next;

Tail->Next = Head;

delete temp;

temp = NULL;

return;

}

if (Tail->RollNo == roll)

{

temp = Head;

while (temp->Next != Tail)

{

temp = temp->Next;

}

delete Tail;

Tail = temp;

Tail->Next = Head;

return;

}

temp = temp->Next;

} while (temp != Head);

}

void Display()

{

Node \*current = Head;

do

{

cout << "Name: " << current->Name << "\nRoll No: " << current->RollNo << "\ncgpa: " << current->Cgpa << endl;

current = current->Next;

} while (current != Head);

}

};

int main()

{

LinkedList list;

list.Insertathead("Noman", "21F-9513", 2.5);

list.Insertathead("Ali", "21F-9511", 2.1);

list.Insertathead("faraz", "21F-9648", 3.6);

list.Display();

list.Insertatlast("Rehan", "21F-9458", 2.8);

list.Insertatlast("Sheraz", "21F-9143", 3.4);

list.Insertatlast("Imran", "21F-9365", 3.6);

list.Display();

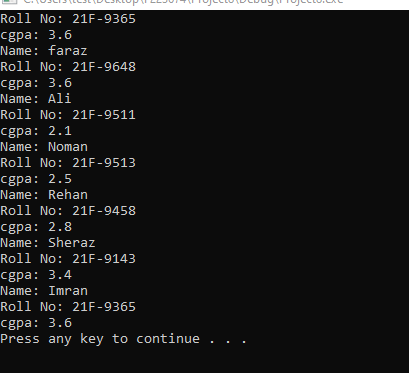
list.Delete("21F-9513");

list.Display();

system("pause");

}

Output:



Task #2:

#include<iostream>

using namespace std;

#define n 10

class Stack

{

int \*arr;

int top;

public:

Stack()

{

arr = new int[n]; //Taking dynamic array

top = -1;

}

void push(int x) //operations of stack

{

if (top==n-1)

{

cout << "Stack over Flow" << endl;

return;

}

top++;

arr[top] = x;

}

void pop()

{

if (top==-1)

{

cout << "No element to pop" << endl;

return;

}

top--;

}

int Top()

{

if (top == -1)

{

cout << "No element in Stack " << endl;

return -1;

}

return arr[top];

}

bool empty()

{

return top == -1;

}

};

int main()

{

Stack obj;

obj.push(1);

obj.push(2);

obj.push(3);

cout << obj.Top() << endl;

obj.pop();

cout << obj.Top() << endl;

obj.pop();

obj.pop();

obj.pop();

cout << obj.empty() << endl;

system("pause");

}

Output:

