Name

Usama Iftikhar Butt

Roll no

17114119-005

Class

CS-17

Assignment # 4

1. Program to implement the doubly linked list.

#include "stdafx.h"

#include "iostream"

using namespace std;

struct Node {

int Data;

Node \*Next;

Node \*Prev;

};

Node \*Head;

void Insert\_Node\_at\_Head(int \_Data) {

Node \*NewNode = new Node;

NewNode->Data = \_Data;

NewNode->Next = NULL;

NewNode->Prev = NULL;

if (Head == NULL) {

Head = NewNode;

}

else {

NewNode->Next = Head;

Head->Prev = NewNode;

Head = NewNode;

}

}

void Insert\_Node\_at\_Tail(int \_Data) {

Node \*NewNode = new Node;

NewNode->Data = \_Data;

NewNode->Next = NULL;

NewNode->Prev = NULL;

if (Head == NULL) {

Head = NewNode;

}

else {

Node \*Temp = Head;

while (Temp->Next != NULL) {

Temp = Temp->Next;

}

Temp->Next = NewNode;

NewNode->Prev = Temp;

}

}

void Insert\_Node\_at\_nth\_Position(int \_Data, int n) {

if (n == 1) {

Insert\_Node\_at\_Head(\_Data);

return;

}

Node \*NewNode = new Node;

NewNode->Data = \_Data;

Node \*Temp = Head;

for (int i = 2; i < n; i++) {

Temp = Temp->Next;

}

if (Temp->Next != NULL) {

NewNode->Next = Temp->Next;

NewNode->Prev = Temp;

Temp->Next->Prev = NewNode;

Temp->Next = NewNode;

}

else {

Temp->Next = NewNode;

NewNode->Prev = Temp;

NewNode->Next = NULL;

}

}

void Forward\_Print() {

Node \*Temp = Head;

cout << "Forward List : ";

while (Temp != NULL) {

cout << Temp->Data << " ";

Temp = Temp->Next;

}

cout << endl;

}

void Reverse\_Print() {

cout << "Reverse List : ";

if (Head != NULL) {

Node \*Temp = Head;

while (Temp->Next != NULL) {

Temp = Temp->Next;

}

while (Temp != NULL) {

cout << Temp->Data << " ";

Temp = Temp->Prev;

}

}

cout << endl;

}

void Invert\_Doubly\_Linked\_List(Node \*Temp) {

if (Temp == NULL) { return; }

if (Temp->Next == NULL) {

Node \*Ptr;

Ptr = Temp->Prev;

Temp->Prev = Temp->Next;

Temp->Next = Ptr;

Head = Temp;

return;

}

Invert\_Doubly\_Linked\_List(Temp->Next);

Node \*Ptr;

Ptr = Temp->Prev;

Temp->Prev = Temp->Next;

Temp->Next = Ptr;

}

void main() {

Head = NULL;

Insert\_Node\_at\_Head(4);

Insert\_Node\_at\_Head(2);

Insert\_Node\_at\_Head(1); //List is : 1 2 4

Insert\_Node\_at\_Tail(9);

Insert\_Node\_at\_Tail(7);

Insert\_Node\_at\_Tail(6); //List is : 1 2 4 9 7 6

Insert\_Node\_at\_nth\_Position(5, 1); //List is : 5 1 2 4 9 7 6

Insert\_Node\_at\_nth\_Position(3, 6); //List is : 5 1 2 4 9 3 7 6

Insert\_Node\_at\_nth\_Position(8, 9); //List is : 5 1 2 4 9 3 7 6 8

Forward\_Print();

Reverse\_Print();

Invert\_Doubly\_Linked\_List(Head); //List is : 8 6 7 3 9 4 2 1 5

Forward\_Print();

Reverse\_Print();

}

2. Program to implement Queue using Linked List

#include "stdafx.h"

#include "iostream"

using namespace std;

struct Node {

int Data;

Node \*Next;

};

Node \*Front = NULL;

Node \*Rear = NULL;

bool Is\_Empty() {

if (Front == NULL) { return(true); }

else { return(false); }

}

void Enqueue(int \_Data) {

Node \*Temp = new Node;

Temp->Data = \_Data;

Temp->Next = NULL;

if (Rear == NULL) { Front = Rear = Temp; }

else {

Rear->Next = Temp;

Rear = Temp;

}

}

void Dequeue() {

if (Is\_Empty()) { cout << "Error! Queue is empty." << endl; return; }

Node \*Temp = Front;

if (Front == Rear) { Front = Rear = NULL; }

else { Front = Front->Next; }

delete Temp;

}

void Front\_Fn() {

if (Front == NULL) {

cout << "Error! Queue is empty." << endl;

return;

}

Node \*Temp = Front;

cout << "Data is : " << Temp->Data << endl;

}

void Print() { //This Print Function is only to see Queue whole Data its not the acctual implementation of Queue

Node \*Temp = Front;

cout << "Queue is : ";

while (Temp != NULL) {

cout << Temp->Data << " ";

Temp = Temp->Next;

}

cout << endl;

}

void main() {

Dequeue(); Print(); //Queue is : (Error)

Enqueue(1); Print(); //Queue is : 1

Enqueue(4); Print(); //Queue is : 1 4

Front\_Fn(); Print(); //Queue is : 1 4

Dequeue(); Print(); //Queue is : 4

Dequeue(); Print(); //Queue is :

Dequeue(); Print(); //Queue is : (Error)

}