**Implement Queue using Double Link List**

#include <iostream>

struct node

{

int data{};

node\* right{nullptr};

node\* left{nullptr};

};

class queue

{

private:

node\* head{nullptr};

public:

bool empty() {return head == nullptr;}

node\* search(int key)

{

node\* temp{head};

while(temp != nullptr)

{

if(temp->data == key)

return temp;

temp = temp->right;

}

return nullptr;

}

void insertAtEnd(int val)

{

node\* newer{new node};

newer->data = val;

if(empty())

{

head = newer;

return;

}

node\* temp{head};

while(temp->right != nullptr)

temp = temp->right;

temp->right = newer; //newer right is nullptr by default

newer->left = temp;

}

void delAtHead()

{

if(empty())

{

std::cout << "empty!\n";

return;

}

node\* temp{head->right};

delete head;

if(temp != nullptr) //if multi nodes,

temp->left = nullptr; //update left of next node(new head)

head = temp; //update head(incase of single node temp would be null)

}

void display()

{

node\* temp = head;

while (temp != nullptr)

{

std::cout << temp->data << " ";

temp = temp->right;

}

std::cout << std::endl;

}

};

int main()

{

queue l{};

l.insertAtEnd(4);

l.insertAtEnd(7);

l.insertAtEnd(98);

l.display();

l.delAtHead();

l.delAtHead();

l.insertAtEnd(3);

l.display();

l.display();

return 0;

}

**Fucn to Display in reverse order**

void reverse()

{

node\* prev{nullptr}, \* curr{head};

while(curr != nullptr)

{

curr->left = curr->right;

curr->right = prev;

prev = curr;

curr = curr->left;

}

head = prev;

}

**Func to count the number of nodes.**

int countNodes() {

int count = 0;

Node\* temp = head;

while (temp != nullptr) {

count++;

temp = temp->next;

}

return count;

**}**

**Middle element of a Doublee**

void findMiddle() {

if (head == nullptr) {

cout << "The list is empty!" << endl;

return;

}

Node\* slow = head;

Node\* fast = head;

while (fast != nullptr && fast->next != nullptr) {

slow = slow->next;

fast = fast->next->next;

**}**

cout << "The middle element is: " << slow->data << endl;

**}**

**Get Nth node in a given Double**

Node\* getNthNode(int n)

{

Node\* temp = head;

int count = 1;

while (temp != nullptr)

{

if (count == n)

{

return temp; // Return the Nth node

}

temp = temp->next;

count++;

**}**

**Insert a new node at any position**

void insertAtPosition(int value, int position) {

Node\* newNode = new Node(value);

Node\* temp = head;

int count = 1;

if (position == 1) {

newNode->next = head;

if (head != nullptr) {

head->prev = newNode;

}

head = newNode;

return;

}

while (temp != nullptr && count < position - 1) {

temp = temp->next;

count++;

}

if (temp == nullptr) {

cout << "The position is greater than the length of the list. Inserting at the end." << endl;

temp = head;

while (temp->next != nullptr) {

temp = temp->next;

}

temp->next = newNode;

newNode->prev = temp;

} else {

newNode->next = temp->next;

newNode->prev = temp;

if (temp->next != nullptr) {

temp->next->prev = newNode;

}

temp->next = newNode;

}

**}**