**WEEK 2B**

**9923103023**

**F1**

Q1)

#include <iostream>

using namespace std;

template<class T>

class stack;

template<class T>

class node{

T data;

node<T> \* next=NULL;

public:

node(T d) {

data=d;

}

friend class stack<T>;

};

template<class T>

class stack{

node<T> \* head=NULL;

node<T> \* tail=NULL;

public:

int len=0;

void push(T data){

len++;

node<T> \* temp=new node<T>(data);

if(!head) {

head=temp;

tail=temp;

return;

}

node<T> \* t=head;

while(t->next!=NULL){

t=t->next;

}

t->next=temp;

tail=temp;

}

void pop(){

if(len==0) return;

node<T> \*temp=head;

len--;

if(temp==tail){

delete temp;

return;

}

while(temp->next!=tail){

temp=temp->next;

}

delete temp->next;

tail=temp;

}

void print(){

node<T> \* temp=head;

while(temp!=NULL){

cout<<temp->data<<' ';

temp=temp->next;

}

cout<<endl;

}

T top(){

return tail->data;

}

};

void check(string s){

stack<char> st;

for(int i=0; i<s.length(); i++){

if(s[i]=='(' || s[i]== '{' || s[i]=='['){

st.push(s[i]);

}

else{

if(s[i]==')' && st.top()=='('){

st.pop();

}

else if(s[i]==']' && st.top()=='['){

st.pop();

}

else if(s[i]=='}' && st.top()=='{'){

st.pop();

}

else{

cout<<"false";

return;

}

}

}

if(st.len==0){

cout<<"true";}

else cout<<"false";

}

int main() {

string s;

cin>>s;

check(s);

return 0;

}

Ans :



Q2)

#include <iostream>

using namespace std;

template<class T>

class stack;

template<class T>

class node {

T data;

node<T>\* next = nullptr;

public:

node(T d) : data(d) {}

friend class stack<T>;

};

template<class T>

class stack {

node<T>\* head = nullptr;

node<T>\* tail = nullptr;

public:

int len = 0;

void push(T data) {

len++;

node<T>\* temp = new node<T>(data);

if (!head) {

head = temp;

tail = temp;

return;

}

tail->next = temp;

tail = temp;

}

void pop() {

if (len == 0) return;

node<T>\* temp = head;

len--;

if (head == tail) {

delete head;

head = tail = nullptr;

return;

}

while (temp->next != tail) {

temp = temp->next;

}

delete tail;

tail = temp;

tail->next = nullptr;

}

void print() const {

node<T>\* temp = head;

while (temp != nullptr) {

cout << temp->data << ' ';

temp = temp->next;

}

cout << endl;

}

T top() const {

if (tail != nullptr) return tail->data;

throw runtime\_error("Stack is empty");

}

bool isEmpty() const {

return len == 0;

}

};

int grtr(stack<int>& s, int target) {

stack<int> tempStack;

stack<int> reversedStack;

int targetPosition = -1;

int nextGreaterPosition = -1;

int index = 0;

bool foundTarget = false;

// Reverse the stack

while (!s.isEmpty()) {

int current = s.top();

s.pop();

reversedStack.push(current);

}

// Find target and the next greater element

while (!reversedStack.isEmpty()) {

int current = reversedStack.top();

reversedStack.pop();

s.push(current);

if (foundTarget) {

if (current > target) {

nextGreaterPosition = index;

break;

}

}

if (current == target) {

foundTarget = true;

targetPosition = index;

}

index++;

}

// Restore the original stack

while (!reversedStack.isEmpty()) {

s.push(reversedStack.top());

reversedStack.pop();

}

if (targetPosition == -1 || nextGreaterPosition == -1) {

cout << "No greater element found to the right of " << target << endl;

return -1; // Return -1 to indicate no valid result

}

return nextGreaterPosition - targetPosition;

}

int main() {

stack<int> s;

int n;

cin>>n;

for(int i=0; i<n; i++){

int a;

cin>>a;

s.push(a);

}

int target;

cin>>target;

int position = grtr(s, target);

if (position != -1) {

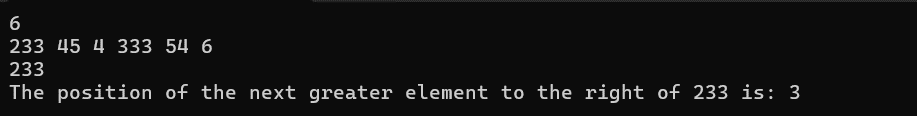
cout << "The position of the next greater element to the right of " << target << " is: " << position

<< endl;

}

return 0;

}



#include <iostream>

using namespace std;

template<class T>

class stack;

template<class T>

class node {

T data;

node<T>\* next = nullptr;

public:

node(T d) : data(d) {}

friend class stack<T>;

};

template<class T>

class stack {

node<T>\* head = nullptr;

node<T>\* tail = nullptr;

public:

int len = 0;

void push(T data) {

len++;

node<T>\* temp = new node<T>(data);

if (!head) {

head = temp;

tail = temp;

return;

}

tail->next = temp;

tail = temp;

}

void pop() {

if (len == 0) return;

node<T>\* temp = head;

len--;

if (head == tail) {

delete head;

head = tail = nullptr;

return;

}

while (temp->next != tail) {

temp = temp->next;

}

delete tail;

tail = temp;

tail->next = nullptr;

}

void print() const {

node<T>\* temp = head;

while (temp != nullptr) {

cout << temp->data << ' ';

temp = temp->next;

}

cout << endl;

}

T top() const {

if (tail != nullptr) return tail->data;

throw runtime\_error("Stack is empty");

}

bool isEmpty() const {

return len == 0;

}

int size() const {

return len;

}

};

int grtr(stack<int>& s, int target) {

if (s.isEmpty()) return -1;

int size = s.size();

int targetPosition = -1;

int nextGreaterPosition = -1;

int index = 0;

bool foundTarget = false;

stack<int> tempStack;

for (int i = 0; i < size \* 2; ++i) {

if (!s.isEmpty()) {

int current = s.top();

s.pop();

tempStack.push(current);

}

if (foundTarget) {

if (tempStack.top() > target) {

nextGreaterPosition = index;

break;

}

}

if (tempStack.top() == target) {

foundTarget = true;

targetPosition = index;

}

index++;

}

while (!tempStack.isEmpty()) {

s.push(tempStack.top());

tempStack.pop();

}

if (targetPosition == -1 || nextGreaterPosition == -1) {

cout << "No greater element found to the right of " << target << endl;

return -1;

}

if (nextGreaterPosition < targetPosition) {

nextGreaterPosition += size;

}

return nextGreaterPosition - targetPosition;

}

int main() {

stack<int> s;

int n;

cin>>n;

for(int i=0; i<n; i++){

int a;

cin>>a;

s.push(a);

}

int target;

cin>>target;

int positionDifference = grtr(s, target);

if (positionDifference != -1) {

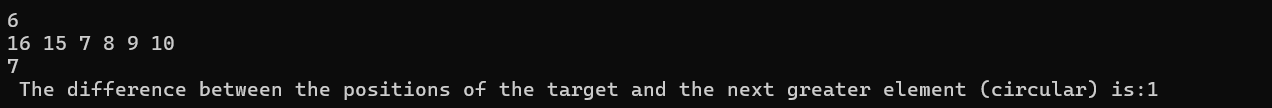
cout << "The difference between the positions of the target and the next greater element

(circular) is: " << positionDifference << endl;

}

return 0;

}



#include <iostream>

#include <unordered\_map>

using namespace std;

template<class T>

class queue;

template<class T>

class node {

T data;

node<T>\* next = nullptr;

public:

node(T d) : data(d) {}

friend class queue<T>;

};

template<class T>

class queue {

node<T>\* head = nullptr;

node<T>\* tail = nullptr;

public:

int len = 0;

void push(T data) {

len++;

node<T>\* temp = new node<T>(data);

if (!head) {

head = temp;

tail = temp;

return;

}

tail->next = temp;

tail = temp;

}

void pop() {

if (len == 0) return;

node<T>\* temp = head;

len--;

if (head == tail) {

delete head;

head = tail = nullptr;

return;

}

head = head->next;

delete temp;

}

void print() const {

node<T>\* temp = head;

while (temp != nullptr) {

cout << temp->data << ' ';

temp = temp->next;

}

cout << endl;

}

T top() const {

if (head != nullptr) return head->data;

throw runtime\_error("Queue is empty");

}

bool isEmpty() const {

return len == 0;

}

int size() const {

return len;

}

};

int findFirstNonRepeatingCharIndex(const string& s) {

unordered\_map<char, int> charFrequency;

queue<char> charQueue;

for (char c : s) {

charFrequency[c]++;

if (charFrequency[c] == 1) {

charQueue.push(c);

} else {

while (!charQueue.isEmpty() && charFrequency[charQueue.top()] > 1) {

charQueue.pop();

}

}

}

if (!charQueue.isEmpty()) {

char firstNonRepeatingChar = charQueue.top();

for (int i = 0; i < s.length(); ++i) {

if (s[i] == firstNonRepeatingChar) {

return i;

}

}

}

return -1;

}

int main() {

queue<char> q;

string s;

cin >> s;

for (char c : s) {

q.push(c);

}

int index = findFirstNonRepeatingCharIndex(s);

if (index != -1) {

cout << "The first non-repeating character is at index: " << index << endl;

} else {

cout << "No non-repeating character found." << endl;

}

return 0;

}

Ans :

