# 15B17CI371 – Data Structures Lab

# ODD 2024

# Week 2-LAB B

# Practice Lab

**1. Given a string ‘s’ containing just the characters '(',')','{','}','[' and ']',determine if the input string is valid. Assume that the string can contain parentheses only i.e.,only '()[]{}' characters are allowed;and the maximum length of string can be,say,20 characters.**

**An input string is valid if:**

**1. Open brackets must be closed by the same type of brackets.**

**2. Open brackets must be closed in the correct order.**

**3. Every close bracket has a corresponding open bracket of the same type.**

#include <iostream>

#include <stack>

using namespace std;

bool isOpeningSymbol(char c)

{

return c=='('||c=='['||c=='{';

}

bool isClosingSymbol(char c)

{

return c==')'||c==']'||c=='}';

}

bool isMatchingPair(char open,char close)

{

return(open=='('&&close==')')||(open=='['&&close==']')||(open=='{'&&close=='}');

}

bool areSymbolsBalanced(string s)

{

stack<char>st;

for(char c:s)

{

if(isOpeningSymbol(c))

st.push(c);

else if(isClosingSymbol(c))

{

if(st.empty()||!isMatchingPair(st.top(),c))

return false;

st.pop();

}

else

return false;

}

return st.empty();

}

int main()

{

string s;

cout<<"Enter a string containing symbols((),[],{}): ";

cin>>s;

if(areSymbolsBalanced(s))

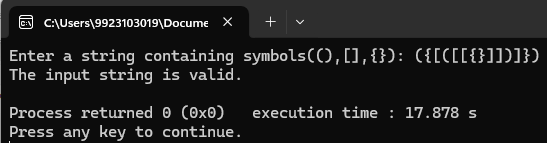
cout<<"The input string is valid."<<endl;

else

cout<<"The input string is not valid."<<endl;

}

**Output :**



**2. Given an array of numbers,input one number from this array and find if it’s next-greater-element exists to the right of this value in the array. If yes,give the position,else print “Not found”. Use stacks to perform this operation.**

#include <iostream>

#include <vector>

#include <stack>

#include <unordered\_map>

using namespace std;

void findNextGreaterPosition(const vector<int>& arr,int element)

{

stack<int> s;

unordered\_map<int,int> nextGreater;

for (int i=arr.size()-1;i>=0;i--)

{

while (!s.empty()&&arr[i]>=s.top())

s.pop();

if (!s.empty())

nextGreater[arr[i]]=s.top();

else

nextGreater[arr[i]]=-1;

s.push(arr[i]);

}

bool found=false;

for (int i=0;i<arr.size();i++)

if (arr[i]==element)

{

int next=nextGreater[element];

if (next!=-1)

for (int j=i+1;j<arr.size();++j)

if (arr[j]==next)

{

cout<<"Element "<<next<<" is the first next-greater-element which exists to the right\n"

<<"of the given element in the array, and it is "<<j-i<<" positions far from "<<element<<endl;

found=true;

break;

}

break;

}

if (!found)

cout<<"Not found"<<endl;

}

int main()

{

int n,num;

cout<<"Input the number of elements: ";

cin>>n;

vector<int> arr;

cout<<"Input the elements: ";

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

{

cin>>num;

arr.push\_back(num);

}

int element;

cout<<"Enter the number to find the next greater element for: ";

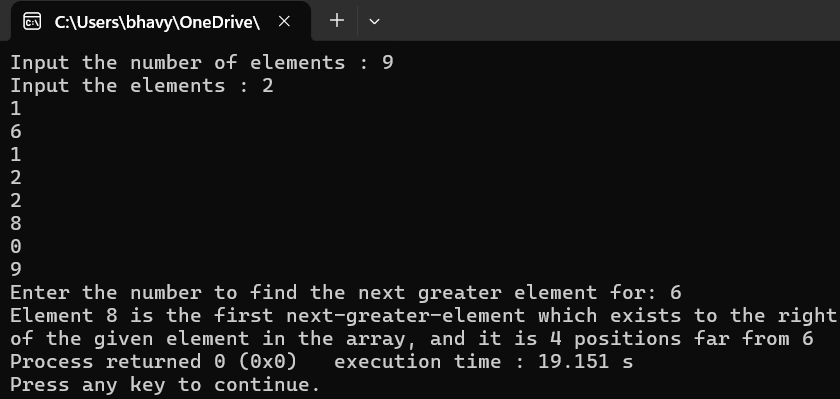
cin>>element;

findNextGreaterPosition(arr,element);

return 0;

}

**Output :**



**3. Modify the above question to allow for circular search.**

#include <iostream>

#include <vector>

#include <stack>

#include <unordered\_map>

using namespace std;

void findNextGreaterPosition(const vector<int>& arr,int element)

{

int n=arr.size();

stack<int> s;

unordered\_map<int,int> nextGreater;

for (int i=2\*n-1;i>=0;i--)

{

int index=i%n;

while (!s.empty()&&arr[index]>=s.top())

s.pop();

if (!s.empty())

nextGreater[arr[index]]=s.top();

else

nextGreater[arr[index]]=-1;

s.push(arr[index]);

}

bool found=false;

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

if (arr[i]==element)

{

int next=nextGreater[element];

if (next!=-1)

for (int j=1;j<n;++j)

{

int index=(i+j)%n;

if (arr[index]==next)

{

cout<<"Element "<<next<<" is the first next-greater-element which is "<<j<<" positions far from "<<element<<endl;

found=true;

break;

}

}

break;

}

if (!found)

cout<<"Not found"<<endl;

}

int main()

{

int n,num;

cout<<"Input the number of elements: ";

cin>>n;

vector<int> arr;

cout<<"Input the elements: ";

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

{

cin>>num;

arr.push\_back(num);

}

int element;

cout<<"Enter the number to find the next greater element for: ";

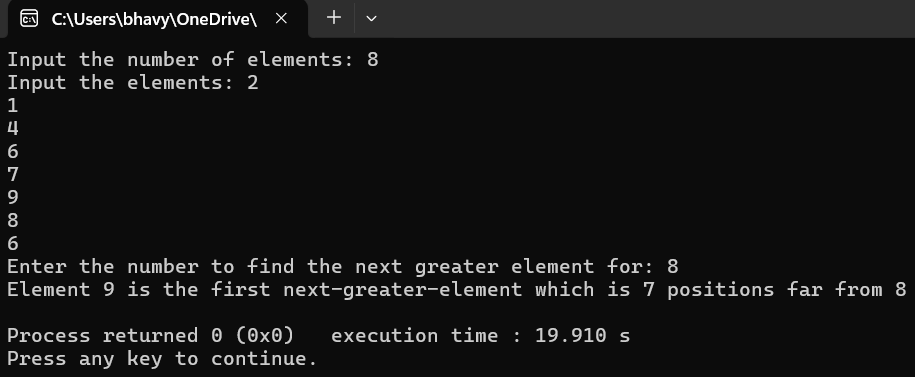
cin>>element;

findNextGreaterPosition(arr,element);

return 0;

}

**Output :**

****

**4. You are given a string ‘s’,find the first non-repeating character (character that occurs only once) in it and return its index (position). If it does not exist,return -1. Implement using concepts of queues.**

#include <iostream>

#include <queue>

#include <unordered\_map>

using namespace std;

int firstNonRepeatingChar(string s)

{

unordered\_map<char,int> count;

queue<int> q;

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

{

count[s[i]]++;

q.push(i);

}

while (!q.empty())

{

int index=q.front();

q.pop();

if (count[s[index]]==1)

return index;

}

return -1;

}

int main()

{

string s;

cout<<"Enter the string: ";

cin>>s;

int index=firstNonRepeatingChar(s);

if (index!=-1)

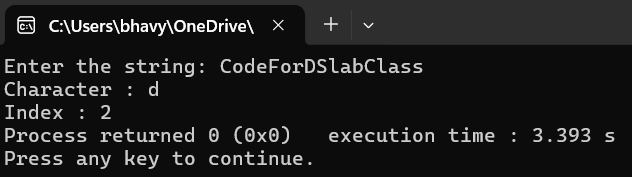
cout<<"Character : "<<s[index]<<endl<<"Index : "<<index;

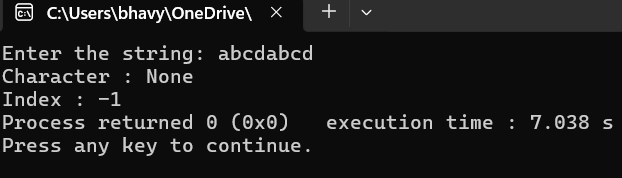
else

cout<<"Character : None"<<endl<<"Index : "<<index;

}

**Output :**

****

****