**Laboratory work #4.**

Solve these problems using stack, queue, deque data structures.

Deadline: week 5

<https://informatics.msk.ru/mod/statements/view.php?id=207#1>

<https://informatics.msk.ru/mod/statements/view.php?id=207&chapterid=55#1>

<https://informatics.msk.ru/mod/statements/view.php?id=207&chapterid=57#1>

<https://informatics.msk.ru/mod/statements/view.php?id=207&chapterid=58#1>

<https://informatics.msk.ru/mod/statements/view.php?id=207&chapterid=60#1>

<https://informatics.msk.ru/mod/statements/view.php?id=207&chapterid=61#1>

<https://informatics.msk.ru/mod/statements/view.php?id=206#1>

// Example program

#include <iostream>

#include <string>

#include <map>

#include <queue>

using namespace std;

int main()

{ int n;

string s;

map <int, queue <string>> mp;

while (cin>>n>>s){

mp[n].push(s);

}

for (auto n: mp) {

while(!n.second.empty()) {

cout<< n.first << " " << n.second.front() << endl;

n.second.pop();

}

}

}

<https://informatics.msk.ru/mod/statements/view.php?id=206&chapterid=50#1>

// Example program

#include <iostream>

#include <string>

#include <stack>

using namespace std;

char getpair(char x){

if (x == '(')

return ')';

if (x == '[')

return ']';

return '}';

}

int main()

{ stack <char> st;

string s;

cin>>s;

for (char x: s){

if (x == '(' || x == '[' || x == '{')

st.push(getpair(x));

else if (st.empty() || st.top() != x){

break;

}

else

st.pop();

}

if (st.empty() ){

cout<<"yes";

}

else

cout<<"no";

}

[https://informatics.msk.ru/mod/statements/view.php?id=206&chapterid=51#1](https://informatics.msk.ru/mod/statements/view.php?id=206&chapterid=51" \l "1)

<https://informatics.msk.ru/mod/statements/view.php?id=206&chapterid=52#1>

<https://informatics.msk.ru/mod/statements/view.php?id=206&chapterid=112984#1>

<https://informatics.msk.ru/mod/statements/view.php?id=206&chapterid=53#1>

<https://leetcode.com/problems/valid-parentheses/>

class Solution {

public:

char pair(char x){

if (x == '(')

return ')';

if (x == '[')

return ']';

return '}';

}

bool isValid(string s) {

stack <char> st;

for (char x: s){

if (x == '(' || x == '[' || x == '{')

st.push(pair(x));

else if (st.empty() || st.top() != x){

return false;

}

else

st.pop();

}

if (st.empty() ){

return true;

}

else

return false;

}

};

<https://leetcode.com/problems/min-stack/>

class MinStack {

public:

/\*\* initialize your data structure here. \*/

int a[9999], sz;

MinStack() {

sz=0;

}

void push(int x) {

a[sz++] = x;

}

void pop() {

sz--;

}

int top() {

return a[sz-1];

}

int getMin() {

int min=a[0];

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

if (min > a[i])

min = a[i];

}

return min;

}};

<https://leetcode.com/problems/backspace-string-compare/>

class Solution {

public:

bool backspaceCompare(string S, string T) {

stack< char > a,b;

for (char i:S){

if(!a.empty() && i == '#'){

a.pop();

}

else if (i != '#') {

a.push(i);

}

}

for (char i:T){

if(!b.empty() && i == '#'){

b.pop();

}

else if (i != '#'){

b.push(i);

}

}

if (a == b)

return true;

else

return false;

}

};

<https://leetcode.com/problems/evaluate-reverse-polish-notation/>