Thời gian còn lại 0:13:57

Câu hỏi 5

Chính xác

Điểm 1,00 của 1,00

You are keeping score for a basketball game with some new rules. The game consists of several rounds, where the scores of past rounds may affect future rounds' scores.

At the beginning of the game, you start with an empty record. You are given a list of strings **ops**, where **ops[i]** is the operation you must apply to the record, with the following rules:

- A non-negative integer **x (from 0 to 9)** - record a new score of **x**
- **'+'** - Record a new score that is the sum of the previous two scores. It is guaranteed there will always be two previous scores.
- **'D'** - Record a new score that is double the previous score. It is guaranteed there will always be a previous score.
- **'C'** - Invalidate the previous score, removing it from the record. It is guaranteed there will always be a previous score.

Finally, return the sum of all scores in the record.

For example:

ops = "52CD+"

- '5' - add to the record. Record now is [5]
- '2' - add to the record. Record now is [5,2]
- 'C' - invalid the previous score (2). Record now is [5]
- 'D' - Record new score that is double of previous score (5*2). Record now is [5,10]
- '+' - Record a new score that is the sum of the previous two scores. Record now is [5,10,15]

Return the sum: 5+10+15 = 30

For example:

Test	Result
`cout << baseballScore("52CD+");`	30
`cout << baseballScore("524CD9++");`	55

Answer: (penalty regime: 0 %)

Reset answer

```cpp
#include <stack>
#include <string>
int baseballScore(string ops){
/*TODO*/
    std::stack<int> scores;

    for (char op : ops) {
        if (op == 'C') {
            if (!scores.empty()) {
                scores.pop();
            }
        } else if (op == 'D') {
            if (!scores.empty()) {
                scores.push(2 * scores.top());
            }
        } else if (op == '+') {
            if (scores.size() >= 2) {
                int top = scores.top();
                scores.pop();
                int newTop = top + scores.top();
                scores.push(top);
                scores.push(newTop);
            }
        } else {
        }
    }

    int totalScore = 0;
    while (!scores.empty()) {
        totalScore += scores.top();
        scores.pop();
    }

    return totalScore;
}
```

Precheck Kiểm tra

	Test	Expected	Got	
✓	`cout << baseballScore("52CD+");`	30	30	✓
✓	`cout << baseballScore("524CD9++");`	55	55	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 6

Chính xác

Điểm 1,00 của 1,00

Implement all methods in class **Stack** with template type **T**. The description of each method is written as comment in frame code.

```
#ifndef STACK_H
#define STACK_H
#include "DLinkedList.h"
template<class T>
class Stack {
protected:
    DLinkedList<T> list;
public:
    Stack() {}
    void push(T item) ;
    T pop() ;
    T top() ;
    bool empty() ;
    int size() ;
    void clear() ;
};

#endif
```

You can use all methods in class **DLinkedList** without implementing them again. The description of class **DLinkedList** is written as comment in frame code.

```
template <class T>
class DLinkedList
{
public:
    class Node;     //forward declaration
protected:
    Node* head;
    Node* tail;
    int count;
public:
    DLinkedList() ;
    ~DLinkedList();
    void add(const T& e);
    void add(int index, const T& e);
    T removeAt(int index);
    bool removeItem(const T& removeItem);
    bool empty();
    int size();
    void clear();
    T get(int index);
    void set(int index, const T& e);
    int indexOf(const T& item);
    bool contains(const T& item);
};
```

For example:

Test	Result

`Stack<int> stack;`
`cout << stack.empty() << " " << stack.size();`
	`1 0`

`Stack<int> stack;`
 	
`int item[] = { 3, 1, 4, 5, 2, 8, 10, 12 };`
`for (int idx = 0; idx < 8; idx++) stack.push(item[idx]);`
 	
`assert(stack.top() == 12);`
 	
`stack.pop();`
`stack.pop();`
 	
`cout << stack.top();`
	`8`

Answer: (penalty regime: 0 %)

Reset answer

```
void push(T item) {
        list.add(item);
    }

    // Pop and return the top item from the stack
    T pop() {
        if (empty()) {
            throw std::runtime_error("Stack is empty");
        }
        return list.removeAt(list.size() - 1);
    }

    // Return the top item without removing it
    T top() {
        if (empty()) {
            throw std::runtime_error("Stack is empty");
        }
        return list.get(list.size() - 1);
    }

    // Check if the stack is empty
    bool empty() {
        return list.empty();
    }

    // Return the number of elements in the stack
    int size() {
        return list.size();
    }

    // Clear the stack
    void clear() {
        list.clear();
    }
```

Precheck Kiểm tra

	Test	Expected	Got	

✓	`Stack<int> stack;`
	`cout << stack.empty() << " " << stack.size();`
		`1 0`	`1 0`	✓

✓	`Stack<int> stack;`
	
	`int item[] = { 3, 1, 4, 5, 2, 8, 10, 12 };`
	`for (int idx = 0; idx < 8; idx++) stack.push(item[idx]);`
	
	`assert(stack.top() == 12);`
	
	`stack.pop();`
	`stack.pop();`
	
	`cout << stack.top();`
		`8`	`8`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 7

Chính xác

Điểm 1,00 của 1,00

Given an array nums[] of size N having distinct elements, the task is to find the next greater element for each element of the array
Next greater element of an element in the array is the nearest element on the right which is greater than the current element.
If there does not exist a next greater of a element, the next greater element for it is -1

Note: iostream, stack and vector are already included

Constraints:
1 <= nums.length <= 10^5
0 <= nums[i] <= 10^9

Example 1:
Input:
nums = {15, 2, 4, 10}
Output:
{-1, 4, 10, -1}

Example 2:
Input:
nums = {1, 4, 6, 9, 6}
Output:
{4, 6, 9, -1, -1}

For example:

Test	Input	Result

`int N;`
`cin >> N;`
`vector<int> nums(N);`
`for(int i = 0; i < N; i++) cin >> nums[i];`
`vector<int> greaterNums = nextGreater(nums);`
`for(int i : greaterNums)`
`    cout << i << ' ';`
`cout << '\n';`
	`4`
	`15 2 4 10`
		`-1 4 10 -1`

`int N;`
`cin >> N;`
`vector<int> nums(N);`
`for(int i = 0; i < N; i++) cin >> nums[i];`
`vector<int> greaterNums = nextGreater(nums);`
`for(int i : greaterNums)`
`    cout << i << ' ';`
`cout << '\n';`
	`5`
	`1 4 6 9 6`
		`4 6 9 -1 -1`

Answer: (penalty regime: 0 %)

Reset answer

```
// iostream, stack and vector are included
using namespace std;
vector<int> nextGreater(vector<int>& arr){
    int n = arr.size();
    vector<int> result(n,-1);
    stack<int> s;
    for (int i=0;i<n;i++)
    {
        while (!s.empty() && arr[i] > arr[s.top()]) {
            result[s.top()] = arr[i]; // Update the result for the top element
            s.pop(); // Remove the top element from the stack
        }
        s.push(i);
    }
    return result;
}
```

Precheck

Kiểm tra

	Test	Input	Expected	Got	

✓	`int N;`
	`cin >> N;`
	`vector<int> nums(N);`
	`for(int i = 0; i < N; i++) cin >> nums[i];`
	`vector<int> greaterNums = nextGreater(nums);`
	`for(int i : greaterNums)`
	`    cout << i << ' ';`
	`cout << '\n';`
		4
		15 2 4 10
			-1 4 10 -1	-1 4 10 -1	✓

✓	`int N;`
	`cin >> N;`
	`vector<int> nums(N);`
	`for(int i = 0; i < N; i++) cin >> nums[i];`
	`vector<int> greaterNums = nextGreater(nums);`
	`for(int i : greaterNums)`
	`    cout << i << ' ';`
	`cout << '\n';`
		5
		1 4 6 9 6
			4 6 9 -1 -1	4 6 9 -1 -1	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 8

Chính xác

Điểm 1,00 của 1,00

Given string **S** representing a **postfix expression**, the task is to evaluate the expression and find the final value. Operators will only include the basic arithmetic operators like ***, /, + and -**.

Postfix expression: The expression of the form "a b operator" (ab+) i.e., when a pair of operands is followed by an operator.

For example: Given string S is "2 3 1 * + 9 -". If the expression is converted into an infix expression, it will be 2 + (3 * 1) – 9 = 5 – 9 = -4.

Requirement: Write the function to evaluate the value of postfix expression.

For example:

Test	Result
`cout << evaluatePostfix("2 3 1 * + 9 -");`	-4
`cout << evaluatePostfix("100 200 + 2 / 5 * 7 +");`	757

Answer: (penalty regime: 0 %)

Reset answer

```cpp
#include<sstream>
int stringToDouble(const string& str)
{
    stringstream ss(str);
    int result;
    ss >> result;
    return result;
}
bool isOperator(const std::string& token) {
    return (token == "+" || token == "-" || token == "*" || token == "/" || token == "^");
}
bool ischarOperator(const char& token) {
    return (token == '+' || token == '-' || token == '*' || token == '/' || token == '^');
}
int evaluatePostfix(string expr){
    stack<int> operandStack;
    stringstream ss;
    ss << expr;
    string token;
    while(ss >> token)
    {
        if (isOperator(token) && !operandStack.empty()) {
            // Pop two operands from the stack
            double operand2 = operandStack.top();
            operandStack.pop();
            double operand1 = operandStack.top();
            operandStack.pop();

            // Perform the operation and push the result back to the stack
            double result;
            if (token == "+") {
                result = operand1 + operand2;
            } else if (token == "-") {
                result = operand1 - operand2;
            } else if (token == "*") {
                result = operand1 * operand2;
            } else if (token == "/") {
```

Precheck

Kiểm tra

	Test	Expected	Got	
✓	`cout << evaluatePostfix("2 3 1 * + 9 -");`	-4	-4	✓
✓	`cout << evaluatePostfix("100 200 + 2 / 5 * 7 +");`	757	757	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

BÁCH KHOA E-LEARNING

WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

(028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

elearning@hcmut.edu.vn

Copyright 2007-2022 BKEL - Phát triển dựa trên Moodle