



Test3_zad1

locked

by milen_chechev

Problem

Submissions

Leaderboard

Discussions

Реализирайте едносвързан списък, който има следните операции:

1. `add(X,pos)` - добавя числото `X` на позиция `pos` в списъка, като ако няма възможност да се добави на такава позиция(отрицателен индекс или прекалено голям индекс) добавя елемента в края на списъка и извежда след това съобщение на стандартният изход `add_last`.
2. `remove(pos)` - премахва елемента, който е на позиция `pos`, ако няма възможност да се премахне на такава позиция(отрицателен индекс или прекалено голям индекс) не се премахва елемент, а на стандартният изход се извежда текста `remove_failed`
3. `print()` - изкарва на стандартният изход числата от списъка, като след всяко число се принтира символа `#`. При празен списък не се изкарва нищо на стандартният изход.
4. `reverse()` - обръща списъка на обратно, т.е. последният елемент става вече първи,предпоследният втори и т.н.
5. `is_palindrom()` - проверява дали списъка е палиндром(поредица,която се чете еднакво отпред назад и отзад напред), като ако е палиндром принтира на стандартният изход `true`, а ако не е `false`
6. `count(X)` - преброява, колко пъти се среща числото `X` в списъка и извежда резултата на стандартният изход
7. `remove_all(X)`-премахва всички срещания на числото `X` в списъка
8. `group(startPos,endPos)`-сумира елементите между подадените две позиции(включително позициите) и ги замества в масива със сумата им. (Пример: при масив `1,2,3,4,5,6,7,8,9` и `group(2,4)` получаваме списък `1,2,12,6,7,8,9`) Ако позициите не са валидни операцията не се изпълнява, а на стандартният изход се принтира `fail_grouping`

Input Format

всеки тест започва с число `N` показващо броя на тестващите редици с операции. След това ще се подадат `K` на брой операции, като първо се подава числото `K` и след това всяка една операция. Всяка операция е на нов ред, като първо е името на операцията, а след това ако операцията има параметри те се подават с разделител интервал

Constraints

ще бъдат подадени максимум 1 милион операции.

Output Format

Изохода е спрямо указанията по-горе, като изхода от всяка тестова редица от операции се извежда на нов ред. При изкарването на резултатите не принтирайте никакви символи(интервали,табулации или нещо друго), които на са в указаниято.

пример:

Вход:

```
2
3
add 1 0
add 2 1
print
5
add 10 0
add 20 0
add 30 2
remove 1
```

```
print
```

Изход:

```
1#2#
```

```
20#30#
```

Тестовите покриват всяка една функция и може да си тествате функционалността по време на писане като ги стартирате.

Sample Input 0

```
4
4
add 1 0
add 2 1
add 3 2
print
4
add 1 0
add 2 0
add 3 0
print
4
add 1 0
add 2 1
add 3 1
print
4
add 1 2
add 2 2
add 3 2
print
```

Sample Output 0

```
1#2#3#
3#2#1#
1#3#2#
add_lastadd_last1#2#3#
```

Sample Input 1

```
6
5
add 1 0
add 2 1
add 3 2
remove 0
print
5
add 1 0
add 2 0
add 3 0
remove 1
print
5
add 1 0
add 2 1
add 3 1
remove 2
print
5
add 1 2
add 2 2
add 3 2
remove 3
print
8
add 1 2
add 2 2
remove 1
add 3 2
remove 1
add 5 2
remove 0
print
9
```

```

add 1 2
add 2 2
remove 1
remove 0
add 3 2
remove 1
add 5 2
remove 0
print

```

Sample Output 1

```

2#3#
3#1#
1#3#
add_lastadd_lastremove_failed1#2#3#
add_lastadd_lastadd_lastadd_last5#
add_lastadd_lastadd_lastremove_failedadd_last5#

```

Sample Input 2

```

4
5
add 1 0
add 2 1
add 3 2
reverse
print
5
add 1 0
add 2 0
add 3 0
reverse
print
7
add 1 0
add 2 1
add 3 1
reverse
print
reverse
print
6
add 1 2
add 2 2
add 3 2
reverse
reverse
print

```

Sample Output 2

```

3#2#1#
1#2#3#
2#3#1#1#3#2#
add_lastadd_last1#2#3#

```

Sample Input 3

```

4
4
add 1 0
add 2 0
add 3 0
is_palindrom
4
add 1 0
add 1 0
add 1 0
is_palindrom
4
add 1 0
add 2 0
add 1 0
is_palindrom

```

```
6
add 1 0
add 2 0
add 3 0
add 2 0
add 1 0
is_palindrom
```

Sample Output 3

```
false
true
true
true
```

Sample Input 4

```
4
4
add 1 0
add 2 0
add 3 0
count 2
4
add 1 0
add 1 1
add 1 1
count 1
4
add 1 2
add 2 2
add 2 2
count 2
9
add 1 0
add 2 1
add 3 2
add 2 1
add 2 4
count 2
count 1
count 3
count 2
```

Sample Output 4

```
1
3
add_lastadd_last2
3113
```

Sample Input 5

```
4
5
add 1 0
add 2 0
add 3 0
remove_all 2
print
5
add 1 0
add 1 1
add 1 1
remove_all 1
print
5
add 1 2
add 2 2
add 2 2
remove_all 2
print
7
add 1 0
add 2 1
```

```
add 3 2
add 2 1
add 2 4
remove_all 2
print
```

Sample Output 5

```
3#1#
add_lastadd_last1#
1#3#
```

Sample Input 6

```
4
5
add 1 0
add 2 0
add 3 0
group 0 0
print
5
add 1 0
add 2 0
add 3 0
group 0 1
print
5
add 1 0
add 2 0
add 3 0
group 0 2
print
5
add 1 0
add 2 0
add 3 0
group 0 3
print
```

Sample Output 6

```
3#2#1#
5#1#
6#
fail_grouping3#2#1#
```

[f](#) [t](#) [in](#)

Submissions: [161](#)



Max Score: 120

Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

[More](#)

Current Buffer (saved locally, editable)  

C++



```
1 #include <cmath>
2 #include <cstdio>
3 #include <vector>
4 #include <iostream>
5 #include <algorithm>
6
7 using namespace std;
8
9 struct Node {
10     //TODO define and use it for the linkedlist
```

```
11 };
12
13 class LinkedList {
14 public:
15     void add(int val, int pos) {
16         // TODO
17     }
18
19     void print() {
20         //TODO
21     }
22
23     void remove(int pos) {
24         //TODO
25     }
26
27     void reverse() {
28         // TODO
29     }
30
31     void remove_all(int val) {
32         // TODO
33     }
34
35     void group(int a, int b) {
36         // TODO
37     }
38
39     void count(int val) {
40         // TODO
41     }
42
43     void is_palindrome() {
44         // TODO
45     }
46
47     ~LinkedList() {
48         // TODO
49     }
50
51 private:
52     // TODO define your private variables
53 };
54
55 int main() {
56     int cases;
57     std::cin >> cases;
58     for (int i = 0; i < cases; i++) {
59         LinkedList ll;
60         int ops;
61         std::string op;
62         std::cin >> ops;
63         for (int j = 0; j < ops; j++) {
64             std::cin >> op;
65
66             if (op == "count") {
67                 int arg1;
68                 std::cin >> arg1;
69                 ll.count(arg1);
70             }
71
72             if (op == "add") {
73                 int arg1, arg2;
74                 std::cin >> arg1 >> arg2;
75                 ll.add(arg1, arg2);
76             }
77
78             if (op == "print") {
79                 ll.print();
80             }
81
82             if (op == "remove") {
83                 int arg1;
84                 std::cin >> arg1;
```

```
85         ll.remove(arg1);
86     }
87
88     if (op == "reverse") {
89         ll.reverse();
90     }
91
92     if (op == "remove_all") {
93         int arg1;
94         std::cin >> arg1;
95         ll.remove_all(arg1);
96     }
97
98     if (op == "group") {
99         int arg1, arg2;
100         std::cin >> arg1 >> arg2;
101         ll.group(arg1, arg2);
102     }
103
104     if (op == "is_palindrom") {
105         ll.is_palindrome();
106     }
107 }
108
109     std::cout << std::endl;
110
111 }
112 return 0;
113 }
```

Line: 1 Col: 1

 [Upload Code as File](#) ☐ Test against custom input[Run Code](#)[Submit Code](#)