Stacks with Doubly Linked List

Problem | Submissions | Leaderboard | Discussions

In this problem you are supposed to implement a stack of integers using doubly linked list, which can support following operations in **constant time**:

1. Push
   - Push some data on the top of stack.
2. Pop
   - Pop the top element from the stack.
3. Get Top
   - Return the topmost element of the stack.

Using the stack, answer following queries, in **constant time**:

1. Print Minimum
   - Get the minimum of all elements of the stack.
2. Print Sum
   - Print the sum of all elements of the stack.

Note 1: You cannot use array/vectors at any point in the program. If you use array, you cannot get score more than 20% of total, and if you use vectors, you will get 0 score, regardless of number of testcases passing.

Note 2: You are supposed to use your own Doubly Linked List Class for the stack implementation. You must implement your own Doubly Linked List as well as Stack Class. Template for same is given below. Do not modify the private data members present in the template.

Note 3: You may use additional space as required. In fact you can use O(stack-size) additional space if it helps. Further, you are free to add any data member to the node structure, stack as it suits you

```
//You may add other data members as well as members functions which may be private or
public, if required. Also, add proper getter/setters.

class Node
{
    private:
        int data;
        Node* prev, next;
    public:
    Node()
    {
        // Implement Constructor
    }
    Node(int d)
    {
        // Implement Constructor
    }

};

class LinkedList
{
    private:
        Node* head;
        Node* tail;
    public:
    LinkedList()
    {
        // Implement Constructor
    }
    void insert(int x)
    {
        // Implement insert
    }
    void remove()
    {
        // Implement remove
    }
};

class Stack
{
    private:
        LinkedList list_obj;
    public:
    void push(int d){
        //Implement your code here
    }
    int pop(){
        //Implement your code here
    }
    int getTop(){
        //Implement your code here
    }
    int getMin(){
        //Implement your code here
    }
    int getSum(){
        //Implement your code here
    }
};
```

Input Format

1. First line is integer `n` representing the number of queries to be performed.
2. Following `n` lines contain integer code `q` representing the corresponding query to be performed. `q` is,
   - `0` for push.
   - `1` for pop.
   - `2` for print top.
   - `3` for print minimum.
   - `4` for print sum.
3. If `q` is `0`, it is followed by integer `d` representing data to be pushed.

Constraints

1. 1 <= n <= 10^5
2. 0 <= q <= 4
3. -10^6 <= d <= 10^6

Output Format

- if `q` is,
  - `0`, do not print anything
  - `1`, do not print anything, even if stack is empty
  - `2`, print the top value of stack, and if stack is empty, print "stack empty"
  - `3`, print the minimum of all values in stack, and if stack is empty, print "stack empty"
  - `4`, print the sum of all values in stack, and if stack is empty, print `0`

Sample Input 0

```
10
0 -1
2
0 -5
4
3
1
2
0 2
3
4
```

Sample Output 0

```
-1
-6
-5
-1
-1
1
```