Question 1:

Design a stack that supports push, pop, peek and retrieving the minimum element in constant time. Returns -1 is the stack is empty. You must implement a solution with O(1) time complexity for each function.

Input Format

First Line should get the number of operations to be performed on stack. List of Operations to be performed in the Stack. Eg 6
["Stack","push","pop","push","peek","pop"] [0,15,0,25,0,0]
0 indicates no input

Constraints

Accept all integers between 1 to 500

Output Format

Displays the value of pop, peek and minimum value in the stack as a list

Sample Input 0

```
6
```

```
["Stack","push","pop","push","push","minval"]
```

```
[0,12,0,39,90,0]
```

Sample Output 0

```
[12,39]
```

Sample Input 1

```
6
```

```
["Stack","push","pop","push","push","minval"]
```

[0,3,0,5,2,0]

Sample Output 1

[3,2]

Question 2:

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (push, peek, pop, and empty). Operations to be performed are as follows: void push(int x) Pushes element x to the back of the queue. int pop() Removes the element from the front of the queue and returns it. int peek() Returns the element at the front of the queue. boolean empty() Returns true if the queue is empty, false otherwise.

Input Format

size of the stack list of operations and values to be pushed

Constraints

Accept all integers between 1 to 500

Output Format

[Elements in the stack, separated by spaces]

Sample Input 0

7

["Queue","push","push","pop","pop","push","pop"]

[0,51,65,0,0,72,0]

Sample Output 0

[51,65,72]

Sample Input 1

4

["Queue","pop","push","pop"]

[0,0,47,0]

Sample Output 1

[-1,47]

Question 3:

Start browsing from homepage,move forward and backwards to other url. BrowserHistory(string homepage) Initializes datastructure with the homepage of the browser. void visit(string url) Visits url from the current page. It clears up all the forward history. string back(int steps) Move steps back in history. If you can only return x steps in the history and steps > x, you will return only x steps. Return the current url after moving back in history at most steps. string forward(int steps) Move steps forward in history. If you can only forward x steps in the history and steps > x, you will forward only x steps. Return the current url after forwarding in history at most steps.

Input Format

11
["BrowserHistory","visit","visit","visit","back","back","forward","visit","forward","back","back"]
["psgtech.edu","google.com","facebook.com","youtube.com",1,1,1,"linkedin.com",2,2,7]

Constraints

Limit the number of operations to a maximum of 15

Output Format

[null,null,null,null,"facebook.com","google.com","facebook.com",null,"linkedin.com","google.com","psgtech.edu"]

Sample Input 0

11

["BrowserHistory","visit","visit","visit","back","back","forward","visit","forward","back","back"]

["psgtech.edu","google.com","facebook.com","youtube.com",1,1,1,"linkedin.com",2,2,7]

Sample Output 0

[null,null,null,null,"facebook.com","google.com","facebook.com",null,"linkedin.com","google.com","ps
gtech.edu"]

Sample Input 1

8

["BrowserHistory","visit","visit","visit","back","forward","visit","back"]

["psgtech.edu","google.com","facebook.com","youtube.com",1,1,"linkedin.com",5]

Sample Output 1

[null,null,null,null,"facebook.com","youtube.com",null,"psgtech.edu"]

Question 4:

The span of the stock's price in one day is the maximum number of consecutive days (starting from that day and going backward) for which the stock price was less than or equal to the price of that day. For example, if the prices of the stock in the last four days is [7,2,1,2] and the price of the stock today is 2, then the span of today is 4 because starting from today, the price of the stock was less than or equal 2 for 4 consecutive days. Also, if the prices of the stock in the last four days is [7,34,1,2] and the price of the stock today is 8, then the span of today is 3 because starting from today, the price of the stock was less than or equal 8 for 3 consecutive days.

Input Format

Number of values in the list List of values

Constraints

Accept all values as integers

Output Format

a list representing the span for stock each day

Sample Input 0

```
7
[100, 80, 60, 70, 60, 75, 85]
```

Sample Output 0

```
[1,1,1,2,1,4,6]
```

Sample Input 1

```
6
[10, 4, 5, 90, 120, 80]
```

Sample Output 1

```
[1,1,2,4,5,1]
```

Question 5:

Given the sorted rotated array nums of unique elements, return the minimum element of this array. You must write an algorithm that runs in O(log n) time. Suppose an array of length n sorted in ascending order is rotated between 1 and n times. For example, the array nums = [0,1,2,4,5,6,7] might become: [4,5,6,7,0,1,2] if it was rotated 4 times.

[0,1,2,4,5,6,7] if it was rotated 7 times. Notice that rotating an array [a[0], a[1], a[2], ..., a[n-1]] 1 time results in the array [a[n-1], a[0], a[1], a[2], ..., a[n-2]].

Input Format

Number of elements list of all values Eg. 5 [3,4,5,1,2]

Explanation: The original array was [1,2,3,4,5] rotated 3 times.

Constraints

Accept rotated sorted array

Output Format

A single integer representing minimum number in the array

Sample Input 0

```
5
[3,4,5,1,2]
```

Sample Output 0

```
1
```

Sample Input 1

```
7
[4 5 6 7 1 2 3]
```

Sample Output 1

```
1
```

Question 6:

A critical point in a linked list is defined as a local minima. A node is a local minima if the current node has a value strictly smaller than the previous node and the next node. Note that a node can only be a local minima if there exists both a previous node and a next node. Given a linked list head, return an array of length 2 containing [minDistance, maxDistance] where minDistance is the minimum distance between any two distinct critical points and maxDistance is the maximum distance between any two distinct critical points. If there are fewer than two critical points, return [-1, -1].

Input Format

List of elements to be added

Constraints

Accept all integers

Output Format

Array of length 2 representing minimum and maximum distance[min,max]

Sample Input 0

```
[3,1]
```

Sample Output 0

```
[-1,-1]
```

Sample Input 1

```
[2,0,1,0,1]
```

Sample Output 1

```
[2,2]
```

Question 7:

A bracket is considered to be any one of the following characters: `(`, `)`, `{`, `}`, `[`, or `]`.

Two brackets are considered to be a *matched pair* if the an opening bracket (i.e., `(`, `[`, or `{`) occurs to the left of a closing bracket (i.e., `)`, `]`, or `}`) *of the exact same type*. There are three types of matched pairs of brackets: `[]`, `{}`, and `()`.

A matching pair of brackets is *not balanced* if the set of brackets it encloses are not matched. For example, `{[(])}` is not balanced because the contents in between `{` and `}` are not balanced. The pair of square brackets encloses a single, unbalanced opening bracket, `(`, and the pair of parentheses encloses a single, unbalanced closing square bracket, `]`.

By this logic, we say a sequence of brackets is *balanced* if the following conditions are met:

- It contains no unmatched brackets.
- The subset of brackets enclosed within the confines of a matched pair of brackets is also a matched pair of brackets.

Given strings of brackets, determine whether each sequence of brackets is balanced. If a string is balanced, return `YES`. Otherwise, return `NO`.

Function Description

Complete the function *isBalanced* in the editor below.

isBalanced has the following parameter(s):

- *string s*: a string of brackets

Returns

- *string:* either `YES` or `NO`

Input Format

The first line contains a single integer , the number of strings.
Each of the next lines contains a single string , a sequence of brackets.

Constraints

-
- , where is the length of the sequence.
- All chracters in the sequences ∈ { **{**, **}**, **(**, **)**, **[**, **]** }.

Output Format

For each string, return `YES` or `NO`.

Sample Input

```
STDIN          Function
-----          --------
3              n = 3
{[()]}         first s = '{[()]}'
{[(])}         second s = '{[(])}'
{{[[(())]]}}    third s ='{{[[(())]]}}'
```

Sample Output

```
YES
NO
YES
```

Explanation

1. The string `{[()]}` meets both criteria for being a balanced string.
2. The string `{[(])}` is not balanced because the brackets enclosed by the matched pair `{` and `}` are not balanced: `[(])`.
3. The string `{{[[(())]]}}` meets both criteria for being a balanced string.

Question 8:

Alexa has two stacks of non-negative integers, stack and stack where index denotes the top of the stack. Alexa challenges Nick to play the following game:

- In each move, Nick can remove one integer from the top of either stack or stack .
- Nick keeps a running sum of the integers he removes from the two stacks.
- Nick is disqualified from the game if, at any point, his running sum becomes greater than some integer given at the beginning of the game.
- Nick's *final score* is the total number of integers he has removed from the two stacks.

Given , , and for games, find the maximum possible score Nick can achieve.

Example

The maximum number of values Nick can remove is . There are two sets of choices with this result.

1. Remove from with a sum of .
2. Remove from and from with a sum of .

Function Description
Complete the *twoStacks* function in the editor below.

twoStacks has the following parameters: - *int maxSum:* the maximum allowed sum
- *int a[n]:* the first stack
- *int b[m]:* the second stack

Returns
- *int:* the maximum number of selections Nick can make

Input Format

The first line contains an integer, (the number of games). The subsequent lines describe each game in the following format:

1. The first line contains three space-separated integers describing the respective values of (the number of integers in stack), (the number of integers in stack), and (the number that the sum of the integers removed from the two stacks cannot exceed).
2. The second line contains space-separated integers, the respective values of .
3. The third line contains space-separated integers, the respective values of .

Constraints

-
-
-
-

Subtasks

- for of the maximum score.

Sample Input 0

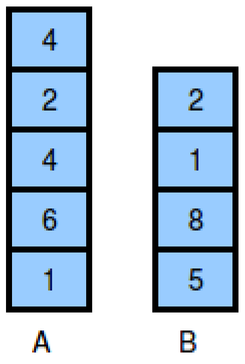
```
1
5 4 10
4 2 4 6 1
2 1 8 5
```

Sample Output 0

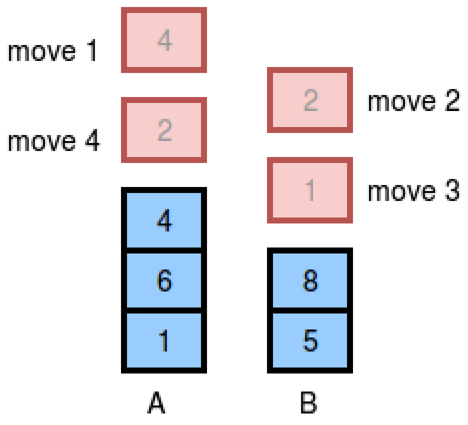
```
4
```

Explanation 0

The two stacks initially look like this:



The image below depicts the integers Nick should choose to remove from the stacks. We print as our answer, because that is the maximum number of integers that can be removed from the two stacks without the sum exceeding .



(There can be multiple ways to remove the integers from the stack, the image shows just one of them.)

Question 9:

Given a reference to the head of a doubly-linked list and an integer, , create a new *DoublyLinkedListNode* object having data value and insert it at the proper location to maintain the sort.

Example

refers to the list

Return a reference to the new list: .

Function Description

Complete the *sortedInsert* function in the editor below.

sortedInsert has two parameters:

- *DoublyLinkedListNode pointer head*: a reference to the head of a doubly-linked list
- *int data*: An integer denoting the value of the field for the *DoublyLinkedListNode* you must insert into the list.

Returns

- *DoublyLinkedListNode pointer:* a reference to the head of the list

Note: Recall that an empty list (i.e., where) and a list with one element *are* sorted lists.

Input Format

The first line contains an integer , the number of test cases.

Each of the test case is in the following format:

- The first line contains an integer , the number of elements in the linked list.
- Each of the next lines contains an integer, the *data* for each node of the linked list.
- The last line contains an integer, , which needs to be inserted into the sorted doubly-linked list.

Sample Input

```
STDIN   Function
-----   --------
1       t = 1
4       n = 4
1       node data values = 1, 3, 4, 10
```

```
3
4
10
5      data = 5
```

Sample Output

```
1 3 4 5 10
```