Implement Stack

- Time limit: 150ms (for almost 16000 pushes and pops)
- Memory limit: 25MB

Implement stack in C# using arrays, you are not allowed to use any other built-in structures (collections , linked-list and etc.).

Your implementation should contains operations below:

- push x: adds element x into top of stack.(x is an int)
- pop: returns the topmost element of stack after removing it from stack.
- peek: returns the topmost element of stack without removing it from stack.
- size: returns number of elements in stack.
- isEmpty: returns true if stack is empty and false if otherwise.

Important : All above operations should operate in O(1) complexity and your stack max size needs to be at least 1000.

Input

```
push 5
push 13
peek
push 12
pop
peek
size
pop
isEmpty
```

```
pop
isEmpty
```

Output

```
13
12
13
2
13
False
5
True
```

Implement Queue

- Time limit: 150ms (for almost 10000 enqueues and dequeues)
- Memory limit: 8MB

Implement Queue in C# using arrays, you are not allowed to use any other built-in structures (collections , linked-list and etc.).

Your implementation should contains operations below:

- enqueue x: adds element x to the end of queue. (x is an int)
- dequeue: returns the element from the front of queue after removing it from queue.
- size: returns number of elements in queue.
- isEmpty: returns true if queue is empty and false if otherwise.

Important : All above operations should operate in O(1) complexity and your queue

max size needs to be at least 1000.

Input

```
enqueue 5
enqueue 13
dequeue
enqueue 12
dequeue
size
isEmpty
dequeue
isEmpty
```

Output

```
5
13
1
False
12
True
```

Implement singly linked list

- Time limit : 700ms (for almost 5000 adds and 5000 appends)
- Memory limit: 25MB

Implement singly linked list in C# , you are not allowed to use built-in structures like collections.

Your implementation should contains operations below:

- add x: adds element x to the beginning of SLL. (x is a String and includes no spaces)
- append x: adds element x to the end of SLL. (x is a String and includes no spaces)
- insert y x: adds element x at before yth element (x is a string and includes no spaces , y is an int and it’s zero-based)
- delete y: delete yth element from SLL.(y is an int and it’s zero-based)
- indexOf x: returns index of first occurrence of element x.(x is a string and includes no spaces)
- printSLL: return list of SLL elements separated by space.

Important : All above operations should operate in O(n) complexity except add x which performs in O(1) .

Input

```
add one
append knocks
insert 1 who
append door
printSLL
indexOf door
```

```
delete 3
printSLL
```

Output

```
one who knocks door
3
one who knocks
```

Implement Queue using two stacks

- Time limit: 150ms (for almost 10000 enqueues and dequeues)
- Memory limit: 15MB

Implement Queue in C# using two stacks, you are not allowed to use any other built-in structures (collections, queue , stack , linked-list and etc.).

You should use the stack you implemented in question 1.

Your implementation should contains operations below:

- enqueue x: adds element x to the end of queue. (x is an int)
- dequeue: returns the element from the front of queue after removing it from queue.
- size: returns number of elements in queue.
- isEmpty: returns true if queue is empty and false if otherwise.

Impotant : All above operations should operate in O(1) complexity and your queue max size needs to be at least 1000.

Input

```
enqueue 5
enqueue 13
dequeue
enqueue 12
dequeue
size
isEmpty
dequeue
isEmpty
```

Output

```
5
13
1
false
12
true
```

Validation

- Time limit: 50ms
- Memory limit: 8MB

Develop a program which takes a string composed of different combinations of '(' , ')', '{', '}', '[', ']' as input and returns a boolean value denoting whether the arrangement is valid or not.

You can use any implemented data structure in questions 1 to 3.

important:

- Expected Time Complexity is O(N) (N is the length of S)
- Expected Auxiliary Space is O(N).
- The length of S is at most 1000.

Input 1

```
()[]{}
```

Output 1

```
1
```

Input 2

```
())({}
```

Output 2

```
0
```

Teris

- Time limit: 50ms
- Memory limit: 8MB

We want to minimize the length of a string. In each minimizing operation, you can remove any two consecutive characters if they are of the form {"as", "to", "by", "vs", "cd", "js", "py", "cs", "cp", "md"}. Develop a program which takes a String as input parameters and returns the the minimum possible length of the string.

Note: you may have to minimize several time.

for instance in ccspk

- First remove cs : ccspk => cpk
- Then remove cp : cpk => k
- Result would be k

You can use any implemented data structure in questions 1 to 3. (though you might need to modify them so that they would accept char as input element)

Important:

- In case of several candidates occurrence choose the rightmost one first.
- Expected Time Complexity is O(N).
- Expected Auxiliary Space is O(N).
- N is at most 1000.

Input

```
atbyvsoszfcpcs
```

Output

```
zf
```

Left-shift SLL

- Time limit: 50ms
- Memory limit: 4MB

You are Given a singly linked list of size N. The task is to left-shift the linked list by k nodes, where k is a given non-negative integer smaller than or equal to length of the linked list.

You should use SLL implemented in question 3.

Important:

- Expected Time Complexity is O(N).
- Expected Auxiliary Space is O(1).
- Linked list max size is 1000.
- Input’s first line contains linked list elements separated by space.
- Input’s second line contains k’s value.

Input

```
2 4 7 8 9
3
```

Output

```
8 9 2 4 7
```

Reverse a sub-list of SLL

- Time limit: 50ms
- Memory limit: 4MB

You are Given a singly linked list of size N and positions m and n. Reverse the linked list from position m to n. m and n are both non-negative and zero-based .

You can use SLL implemented in question 3.

Important:

- Expected Time Complexity is O(N).
- Expected Auxiliary Space is O(1).
- Linked list max size is 1000.
- Input’s first line contains linked list separated by space.
- Input’s second line contains m and n .

Input

```
1 7 5 3 9 8 10 2 2 5
1 8
```

Output

```
1 2 2 10 8 9 3 5 7 5
```