1. Take a string from the user and print it in the reverse order using **stack** data structure. **Also prints the unicode of each character**

Input:

W A T E R M E L O N

Output:

N O L E M R E T A W

Hint: Use the property of stack to reverse the string

2. Given a string of brackets, determine whether each sequence of brackets is balanced. If a string is balanced, print True on a new line;otherwise, print False on a new line.

Input :	Output
()	True
{}	True
[]	True
{[]}	True
{([])}	False
(()	False
{[(]}	False

3. Suppose, you have a stack class. You have 5 utility functions: isempty(), push(), pop(), peek() and size(). Write a program keepLargestOnTop(stack) in a way that the largest value in that stack would be in the TOP position always. Return the stack.
Hints: You should use a helpingStack and its functions helpingStackPush(), helpingStackPop(), helpingStackisEmpty(), helpingStacksize(). Assume: these are already implemented. You need to use them properly.

Current stack

23
53
56
19
44
22

99
44

TOP

After your Program keepLargestOnTop(stack): Stack looks like below:

23
53
56
19
44
22
44
99

TOP

4. Given two stacks with integer values, balance the sum of integer values present in each of the stack by adding a new element in any of the stack. For example, if stack **A** contains 1, 5, 11 and 4 and stack **B** contains 4 ,32,-1 and 1 then by adding 15 in stack **A** will balance the sum of values of boththe stacks.

Input:

1	4
5	32
11	1
4	-1

Stack A Stack B

Output:

1	4
5	32
11	1
4	-1
	15

StackA Stack B

5. We have a stack class and have 5 utility functions: isempty(), push(), pop(), peek() and size(). The stack is maintaining the ascending order always. Now, you would need to insert a new value into this stack using a **pushStack(**value**)** function. Write the **pushStack(value) utility** function for the stack class so that the function always maintains the ascending order in the stack.

Program a simple text editor using Stack. The user will input a complete sentence. You have to put this string into a character stack.When the user selects the option to undo, a word from the stack will be deleted. By selecting undo again, another word from the stack willbe removed and so on.

Hints: Create a stack for words.

6. Reverse the first 'k' elements of a queue, push it back into the queue. For example, if the queue has members 10, 20, 30, 40 , 50 , 60 , 70, 80, 90 and first 5 numbers are asked to be reversed then the result will be 60, 70, 80, 90, 50, 40, 30, 20, 10.

Input:

10	**20**	**30**	**40**	**50**	60	70	80	90

Output:

60	70	80	90	**50**	**40**	**30**	**20**	**10**

7. You have two volunteers who are collecting donations from the members of a community. Suppose each of the members has a member idand they are lined up in ascending order. To keep the number of people balanced, volunteer A is collecting donations from members whose ids aredivisible by 2 and volunteer B is collecting donations from members whose ids are not divisible by 2. Write a program which uses two queues, such that if the even number member id is given, it will assign that member to volunteer A, otherwise, the member will go to volunteer B.

Input:

2 4 7 10 1 9 11

Output:

2	4	10		

Queue A

7	1	9	11	

Queue B

8. Given a number n, write a function that generates and prints all binary numbers with decimal values from 1 to n. You need to use a **queue** for that

Input:5

Output:0

01

10

11

100

101

Input:3

Output:

0

01

10

11

Input:1

Output:

0

01

9. Create a myQueue class and all of it's functions (enqueue, dequeue, isEmpty, peek, size). You can use only two stacks (two objects from your stack class) inside the myQueue class to hold the data. That means, you cannot use any list or double ended queue to hold data.

10. Create a myStack class and all of it's functions (push, pop, isEmpty, peek, size). You can use only two queues (two objects from your queue class) inside the myStack class to hold the data. That means, you cannot use any list to hold data.

11. Given an array of integers temperatures represents the daily temperatures, return an array answer such that answer[i] is the number of days you have to wait after the ith day to get a warmer temperature. If there is no future day for which this is possible, keep answer[i] == 0 instead.

Example 1:

Input: temperatures = [73,74,75,71,69,72,76,73]

Output: [1,1,4,2,1,1,0,0]

Example 2:

Input: temperatures = [30,40,50,60]

Output: [1,1,1,0]

Hints: If the temperature is say, 70 today, then in the future a warmer temperature must be either 71, 72, 73, ..., 99, or 100. We could remember when all of them occur next. You should use a stack to keep the track.