Introduction to Computers and Programming LAB-8 2016/11/23

- ✧ The output must be in our sample output format.
- ✧ You can raise your hand to demo once you finish a program.
- ✧ The bonus question is for this lab only. If you cannot finish question 1 to 4 in time, you are allowed to demo them at next lab hours.
- ✧ TAs will update lab records every Monday after the lab hours in the link: http://goo.gl/ZVJu2Y

1. Substring

Please design a program with a function that can identify whether the 1st input string is the substring of the 2nd input string or not.

Here are the 3 kinds of output that you should output:

(1) "String1 is a Continuous substring of String2"

(2) "String1 is a Non-continuous substring of String2"

(3) "String1 is not a substring of String2"

Note:

(1) ***You can only use <stdio.h> , i.e. you cannot use <string.h>, etc.***

(2) The Continuous substring means every character of string1 is shown in string2 with the correct order and all of them are continuous.

(3) The Non-continuous substring means every character of string1 is shown in string2 with the correct order, and it doesn't matter if all of them are continuous or not.

(4) Your function must can handle white space as a normal character.

```
abc
abdabcabe
String1 is a Continuous substring of String2

abc
abdacabe
String1 is a Non-continuous substring of String2

ab
acdeaccde
String1 is not a substring of String2
```

```
jkl jl
jkfe l jfl fewjkl jlfeww
String1 is a Continuous substring of String2

jkl jl
jkfe l jfl
String1 is a Non-continuous substring of String2

jkl jl
jkfeljfl
String1 is not a substring of String2
```

2. Bubble Sort

Write a program that first reads the size representing the size (<= 10) of following input sequence. Then, it reads a sequence of positive integers. Use **Bubble Sort** to sort and output the integers from the smallest to the largest. Also, implement a function **`void swap (int *p, int *q);`** that exchange the values of the variables in Bubble Sort. Please refer to ***Bubble Sort.gif*** to learn how Bubble Sort works.

```
Input the size: 4
Before sort: 9 5 2 7
After sort: 2 5 7 9

Input the size: 5
Before sort: 1 2 3 4 5
After sort: 1 2 3 4 5

Input the size: 3
Before sort: 1 1 1
After sort: 1 1 1
```

3. Text Reverse 2

Write a program can reverse the text which is encased by parentheses. If there are parentheses inside parentheses, you have to reverse inner encased text first and then reverse outer encased text.

If there is a not pair '(' reverse all the text after it.

If there is a not pair ')' reverse all the text before it.

Note: the length of input is not greater than 100.

e.g.

Input:

a((b(cde)f)gh)i

Process:

a((b(cde)f)gh)i → a((bedcf)gh)i → a(fcdebgh)i → ahgbedcfi

Output:

ahgbedcfi

Input:

apple)(is (red).

Process:

apple)(is (red). → elppa(is (red). → elppa(is der. → elppa.red si

Output:

elppa.red si

```
Please use '(' and ')' to mark reverse area, your input:
(erised)
Result:
desire
```

```
Please use '(' and ')' to mark reverse area, your input:
((apple) (is) (red) (and) (banana) (is) (yellow))
Result:
yellow is banana and red is apple
```

4. Stack

A stack is a special structure of arrays *(Please refer to Chap.10 if you aren't familiar with stacks.)*.

Implement **a non-negative integer stack** with maximum size 5. There are three operations:

(1) ***push(n)***: push value n into the stack. Show message "n is pushed." after the operation. If the stack is full, show "The stack is already full."

(2) ***pop***: pop an element out of the stack. Show message "n is popped." after the operation. If there is no element to pop, show "The stack is already empty."

(3) ***show***: show the element of the stack. The last element represents the last pushed one. If the stack is empty, show "The stack is empty."

The input is a string representing the operation, the output should follow the rules describing above. Also, you should handle with any illegal input operations. Here are examples for illegal operations:

legal	illegal	
push(n)	Left and right parentheses are not in **a pair**	push, push(, push((), push((2))
	The position of parentheses is opposite	push)2(
	There are something behind ')' or before 'push'	pushpush(2), push(2)(3)(4)
	There is other symbols in parenthesis	push(-2), push(3.5)
pop	pop(1), pop3, popular, 123pop	
show	showtime, show5566, s h o w, show(5566)	

```
- push(1)
1 is pushed.
- push(3)
3 is pushed.
- push(2)
2 is pushed.
- push(5
It is an illegal operation.
- push(5)
5 is pushed.
- push(4)
4 is pushed.
- push(6)
The stack is already full.
- show
1 3 2 5 4
- pop
4 is popped.
- pop
5 is popped.
- push(100)
100 is pushed.
- show
1 3 2 100
- ppush(1)
It is an illegal operation.
- push(2.5)
It is an illegal operation.
```

5. (BONUS) Calculator

In this question, you are going to implement a simple text calculator, which is able to deal with a series of arithmetic expression. **Given a valid arithmetic expression, the program output the result; otherwise, message "invalid expression!" should show on the screen**. The followings are the rules of this calculator:

1. Operands in the expression will be integers only, no floating point numbers. However, please output the result in floating point number, and round the result to the third decimal points.
2. There are four binary operators: + (add), - (minus), * (multiply), / (divide). Also, the unary minus could only apply to the first term of number.
3. You don't need to handle with parentheses ( ).
4. The priority of *, / is higher than +, -. If two operations share the same priority, then it will be calculated from left to right.
5. No two continuous operators are allowed.
6. Space and tab characters won't affect value of the number input.

Here are some examples which may make you feel clearer to the above rules.

Valid expression	Invalid expression
1+2+3 1 +2 + 3 2 0 0-1 -1+2*5 3*5-12/4-1/5 100 -100	1--2 7 * - 7 +3 12 / * 5/0

Output format example

(Please round the output to the third digit after the decimal point.)