

[Data Structure HW #4]

Submit C program source file, execution screen capture file. You should check your program at <https://repl.it/languages/c>

아래 모든 문제에서 작성한 파일들 소스파일, 헤더파일 모두를 제출하고, 위의 실행환경에서 실행한 결과를 캡처해서 같이 제출할 것!

All the files (source code, header) for the following problems should be submitted with the screen capture files showing their execution at <https://repl.it/languages/c>.

1. Implement the following ADT stack using **array** and **linked list**; you should **implement both of them**. You should **utilize them at the next problems**.

다음의 ADT를 만족하는 stack을 array와 linked list 두 가지 버전으로 구현하시오. 이 구현한 부분을 이후 문제에서는 활용할 것.

ADT Stack

- **Object: a finite ordered list with zero or more elements**

- **Operations**

- *Stack* **CreateStack(size)** ::= create an empty stack whose maximum size is *size*
- *void* **Push(stack, item)** ::= if *stack* is full, error
else insert *item* into top of *stack*
- *Element* **Pop(stack)** ::= if *stack* is empty, error
else remove and return the item on top of *stack*
- *Element* **Top(stack)** ::= if *stack* is empty, error
else return the item on top of *stack*
- *void* **DestroyStack(stack)** ::= remove all items and deallocate memory
- *Boolean* **IsFullStack(stack)** ::= if stack is full return true
else return false
- *Boolean* **IsEmptyStack(stack)** ::= if stack is empty return true
else return false
- *int* **CountStackItem(stack)** ::= returns # of items in stack
- *void* **ClearStack(stack)** ::= remove all items in stack

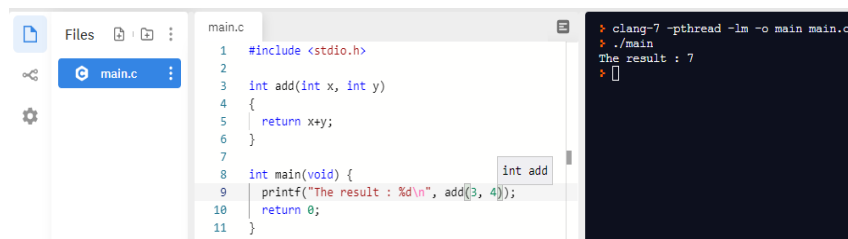
Stack ADT의 array 버전은 **Stack_Array.h** 에 구현하고, Stack ADT의 linked list 버전은 **Stack_Link.h** 에 구현할 것. 이후 문제에서는 이 파일들을 include 해서 프로그램을 작성할 것. 사용예는 아래에 있음.

Stack_Array.h must include the array implementation of Stack ADT.

Stack_Link.h must include the linked list implementation of Stack ADT.

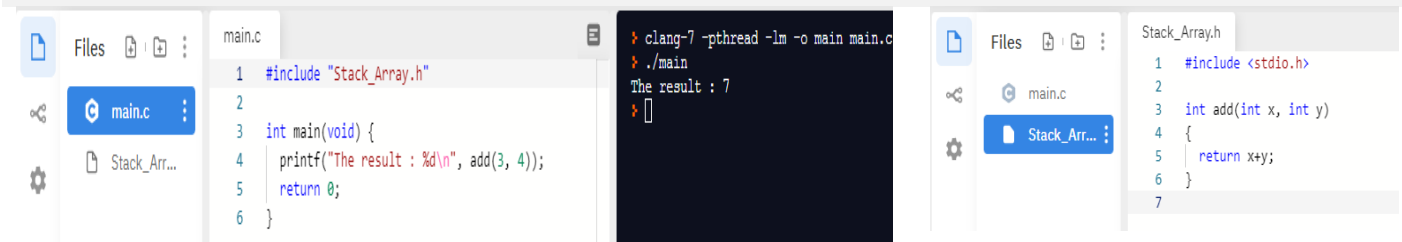
At the following problems, you should utilize them at the main() function by #include "Stack_Array.h" or #include "Stack_Link.h"

<Use Example>



```
main.c
1 #include <stdio.h>
2
3 int add(int x, int y)
4 {
5     return x+y;
6 }
7
8 int main(void) {
9     printf("The result : %d\n", add(3, 4));
10    return 0;
11 }
```

```
> clang-7 -pthread -lm -o main main.c
> ./main
The result : 7
> []
```



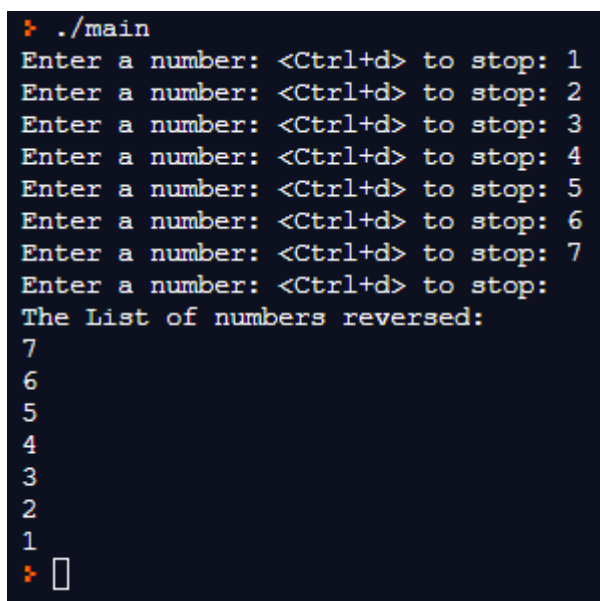
```
main.c
1 #include "Stack_Array.h"
2
3 int main(void) {
4     printf("The result : %d\n", add(3, 4));
5     return 0;
6 }
```

```
Stack_Array.h
1 #include <stdio.h>
2
3 int add(int x, int y)
4 {
5     return x+y;
6 }
7
```

```
> clang-7 -pthread -lm -o main main.c
> ./main
The result : 7
> []
```

2. Reversing data. Print out the input data as its reversing order. : Source file name : **Reverse.c**

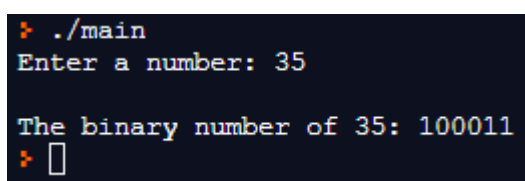
Execution Example :



```
> ./main
Enter a number: <Ctrl+d> to stop: 1
Enter a number: <Ctrl+d> to stop: 2
Enter a number: <Ctrl+d> to stop: 3
Enter a number: <Ctrl+d> to stop: 4
Enter a number: <Ctrl+d> to stop: 5
Enter a number: <Ctrl+d> to stop: 6
Enter a number: <Ctrl+d> to stop: 7
Enter a number: <Ctrl+d> to stop:
The List of numbers reversed:
7
6
5
4
3
2
1
> []
```

3. Convert decimal to binary. If you get a positive integer as input, you should print out its binary number. Source file name : **Convert.c**

Execution Example:



```
> ./main
Enter a number: 35

The binary number of 35: 100011
> []
```

4. 괄호를 포함하는 4칙 연산(+, -, *, %)이 가능한 Calculator. 다음을 포함해야 함. Source file name : **Calculator.c**

Implement a calculator which can add, subtract, multiply and modulate (+, -, *, %). But, your program should meet the following requirements.

- 양의 정수 수식을 토큰으로 받아서 token으로 분석 (빈칸으로 구분이 안되는 붙은 식도 분석 가능해야 함. C에서 제공하는 token 분석하는 함수도 있음.) 빈칸도 있을 수 있고, 괄호도 있을 수 있고, 두 자리 이상의 수도 처리가능해야 함. 단, 식의 계산 결과는 양수로 가정. (음수 처리, 분수 처리 없음)

Your program should analyze an expression given (positive integers) by token. The expressions will contain space, '(', and ')' as well as double or three figures. Maybe, there is no space at the expression given. The calculation results are assumed to be positive integer.(No negative number, No fraction) You can utilize the token analyzer functions in C language.

- Infix notation ➔ postfix 로 변환

Your program should change infix notation (the expression given) to postfix notation.

- 수식의 유효성 검사. 예를 들면, 괄호를 열기만 하고 닫지 않았다던지 하는부분 및 이항연산식을 표현한 건지 등.

Your program should check the validity of the expression given. For example, the pair of (), whether the expression can be calculated or not.

- 이 문제를 위해서 1번에서 작성한 "Stack_Array.h" 와 "Stack_Link.h"를 변형해야 할 일이 있으면, 변형된 버전은 "**Stack_Array2.h**" 와 "**Stack_Link2.h**" 로 이름지어서 활용하고 과제 제출시 제출할 것.

You may modify "Stack_Array.h" and "Stack_Link.h" for this problem. If you did, please let the name of them be "**Stack_Array2.h**" and "**Stack_Link2.h**" each.

Execution Sample:

```
❯ ./main
Arithmetic Expression : 3 * (4- 3) % 3
Input : 3 * (4- 3) % 3
Post : 3 4 3 - * 3 %
Result : 0
❯ □

❯ ./main
Arithmetic Expression : 3 * ( ( 67 - 60)*4 - 20 )
Input : 3 * ( ( 67 - 60)*4 - 20 )
Post : 3 67 60 - 4 * 20 - *
Result : 24
❯ □

❯ ./main
Arithmetic Expression : 2*(54-50)+8*(9%4)
Input : 2*(54-50)+8*(9%4)
Post : 2 54 50 - * 8 9 4 % * +
Result : 16
❯ □
```

```
❖ ./main
Arithmetic Expression : 34 56 - 7 *8
Error : Check the binary expression !
❖
```

```
❖ ./main
Arithmetic Expression : 3 - ( ( 5 + 61 ) * 7
Error : Check the pairs of Parenthesis!
❖
```