LAB4

Lab4

- The deadline for lab4 submission is 29th March at 11:59 pm.

- Folder name : lab4
- Code name: p4.c
- Each code will be tested by 5 different input files.
- 5 score for each input, if you don't get the answer, you get 0 score.

Evaluation criteria

Category	Evaluation	
p4	100	
Total	100	

- *Use GCC **11** version.*
- *No score will be given if the gcc version is different.*

Lab 4. Postfix evaluation

$$7 \;\; \underbrace{2 \;\; 3 \;\; *}_{2 * 3 \,=\, 6} \;\; - \;\; 4 \;\; \% \;\; 9 \;\; 3 \;\; / \;\; +$$

$$\underbrace{7 \;\; 6 \;\; -}_{7 - 6 \,=\, 1} \;\; 4 \;\; \% \;\; 9 \;\; 3 \;\; / \;\; +$$

$$\underbrace{1 \;\; 4 \;\; \%}_{1 \,\%\, 4 \,=\, 1} \;\; 9 \;\; 3 \;\; / \;\; +$$

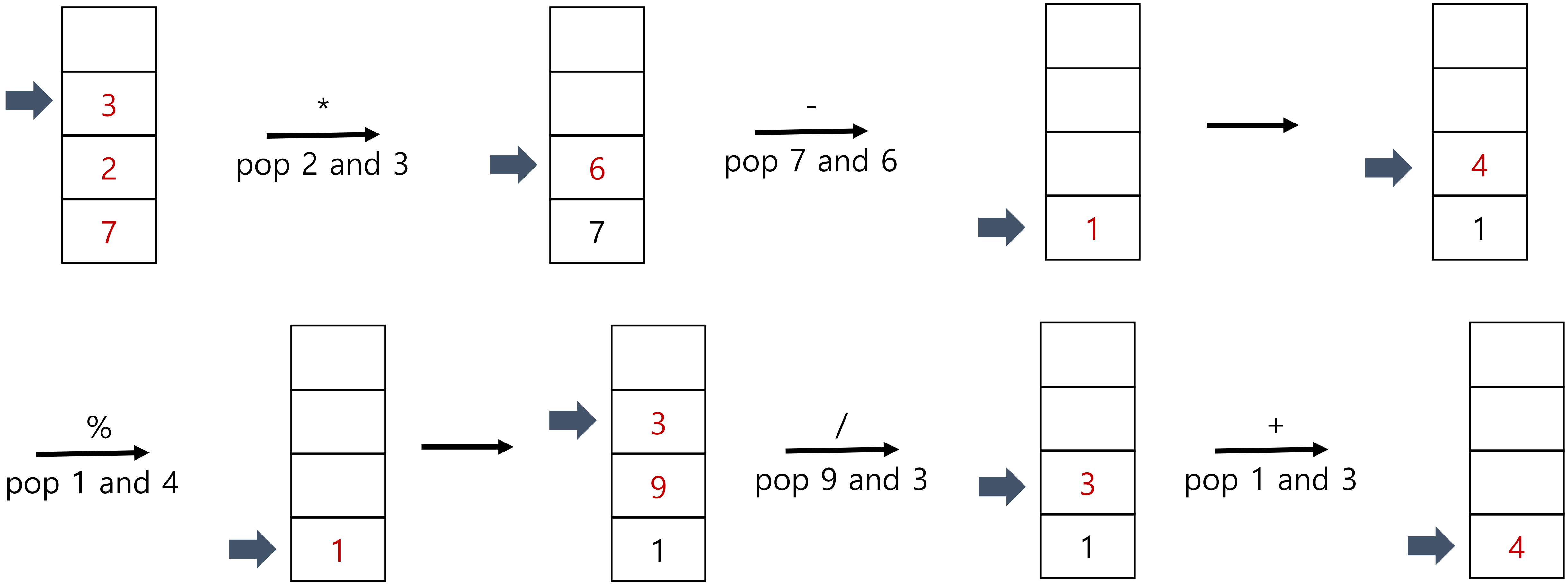
$$1 \;\; \underbrace{9 \;\; 3 \;\; /}_{9 / 3 \,=\, 3} \;\; +$$

$$\underbrace{1 \;\; 3 \;\; +}_{1 + 3 \,=\, 4}$$

Lab 4. Postfix evaluation using Stack

7 2 3 * - 4 % 9 3 / +

➡ : Top of stack



Lab 4. Postfix evaluation using Stack

- Available operators: +, -, *, /, and %
- Not used: (,)
- Operands: single-digit numbers (1, 2, 3, 4, 5, 6, 7, 8, and 9)
- Conditions:
  - The expression should be no more than 100 characters.
  - The delimiter for the end of the expression is ‘#’.
  - Max stack size is 10.
- There are two rules for popping and pushing the operands from/to the stack:
  - When you meet an operand (number), push it onto the stack.
  - When you meet an operator, pop two operands from the stack, perform the operation, and push the result back to the stack.

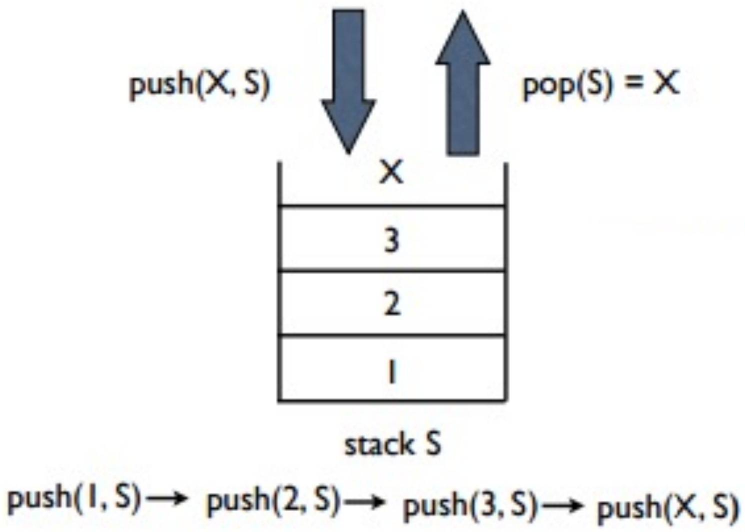
Lab 4. Postfix evaluation using Stack

- **CreateStack** create a new stack with the size of max.
- **Push** push a new element in the stack. If your stack is full, just print an error message and exit the program.
- **Pop** pop the element from the top of the stack. If the stack does not have any element, just print an error message and exit the program.
- **DeleteStack** free all the memory allocated to stack.
- **IsFull** check if the stack is full.
- **IsEmpty** check if the stack is empty.
- **Postfix** When you meet an operand (number), push it onto the stack. When you meet an operator, pop two operands from the stack, perform the operation, and push the result back to the stack.

Lab 4. Postfix evaluation using Stack

- Structure

```
typedef struct Stack{
        int* key;
        int top;
        int max_stack_size;
}Stack;
```



Lab 4. Postfix evaluation using Stack

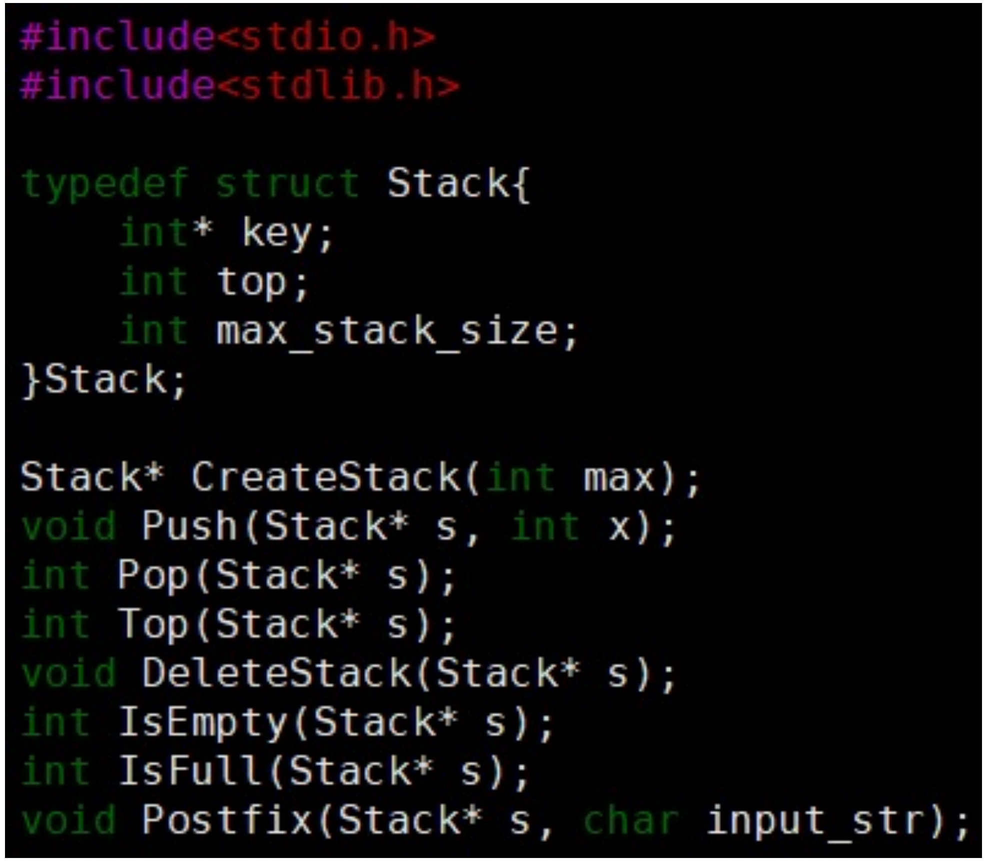
- Structure

```
typedef struct Stack{
        int* key;
        int top;
        int max_stack_size;
}Stack;
```

- Function

```
Stack* CreateStack(int max);
void Push(Stack* s, int x);
int Pop(Stack* s);
int Top(Stack* s);
void DeleteStack(Stack* s);
int IsEmpty(Stack *s);
int IsFull(Stack *s);
void Postfix(Stack* s, char input_str);
```

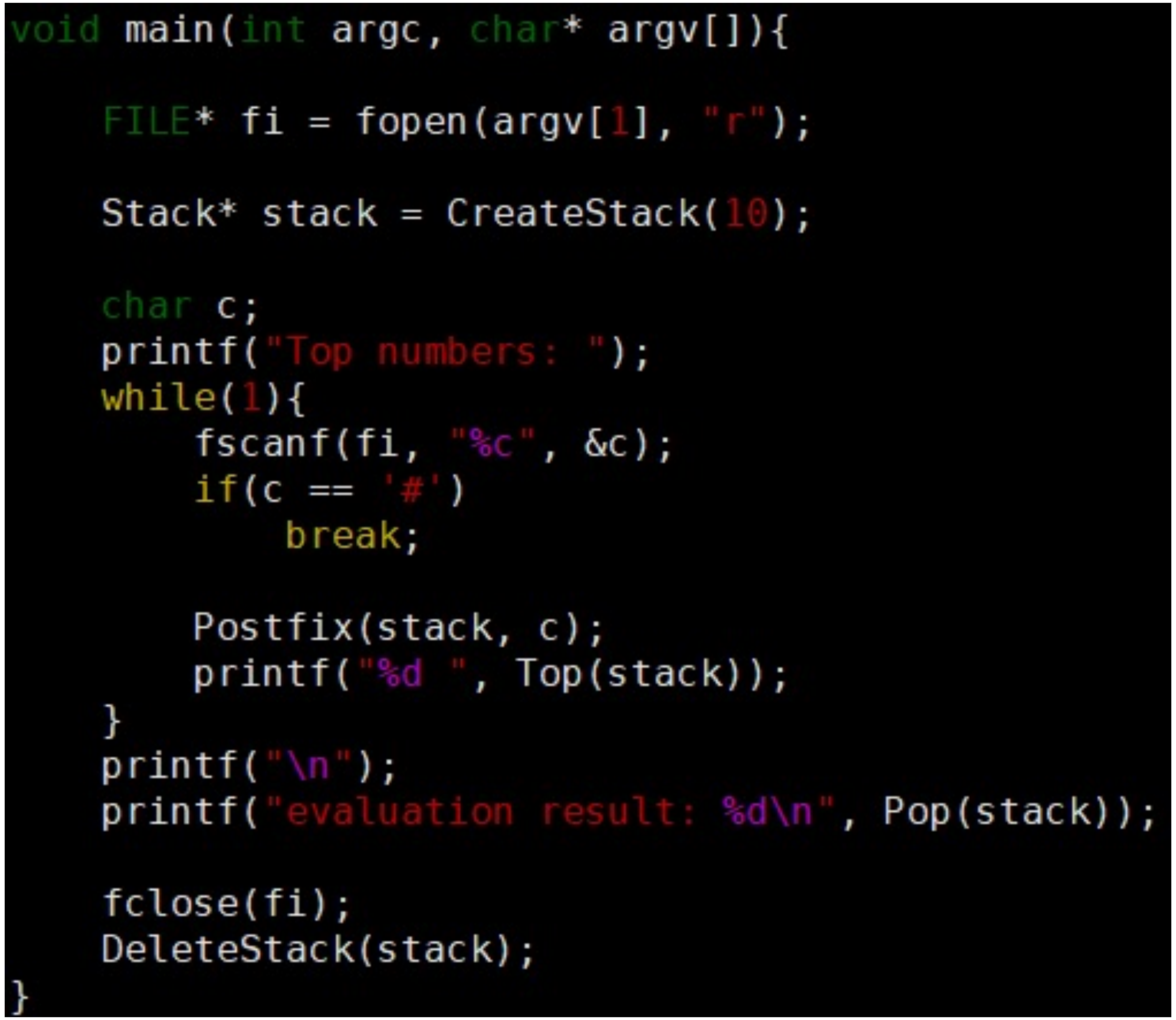
Lab 4. Sample code



```c
#include<stdio.h>
#include<stdlib.h>

typedef struct Stack{
    int* key;
    int top;
    int max_stack_size;
}Stack;

Stack* CreateStack(int max);
void Push(Stack* s, int x);
int Pop(Stack* s);
int Top(Stack* s);
void DeleteStack(Stack* s);
int IsEmpty(Stack* s);
int IsFull(Stack* s);
void Postfix(Stack* s, char input_str);
```



```c
void main(int argc, char* argv[]){

    FILE* fi = fopen(argv[1], "r");

    Stack* stack = CreateStack(10);

    char c;
    printf("Top numbers: ");
    while(1){
        fscanf(fi, "%c", &c);
        if(c == '#')
            break;

        Postfix(stack, c);
        printf("%d ", Top(stack));
    }
    printf("\n");
    printf("evaluation result: %d\n", Pop(stack));

    fclose(fi);
    DeleteStack(stack);
}
```

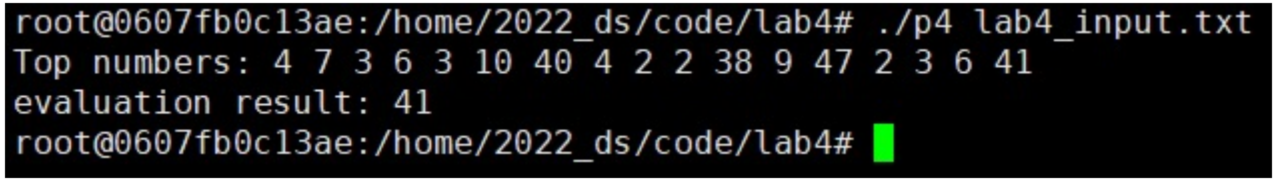
Lab 4. Example

- input file: lab4_input.txt

```
4736%+*42/-9+23*-#
```

- Result

  every time there is a push, print out the top number



```
root@0607fb0c13ae:/home/2022_ds/code/lab4# ./p4 lab4_input.txt
Top numbers: 4 7 3 6 3 10 40 4 2 2 38 9 47 2 3 6 41
evaluation result: 41
root@0607fb0c13ae:/home/2022_ds/code/lab4# 
```

Lab 4. Postfix evaluation using Stack

- program name : p4.c
- input : a list of operations in a file.
- output : the corresponding result in the standard output.