Lab 4 & HW 4

Data Structure

Lab 4 (due on Lab Session)

1. Do p4_1.c

HW 4 (due on the day before the next Lab Session)

1. Do p4_2.c

Evaluation criteria

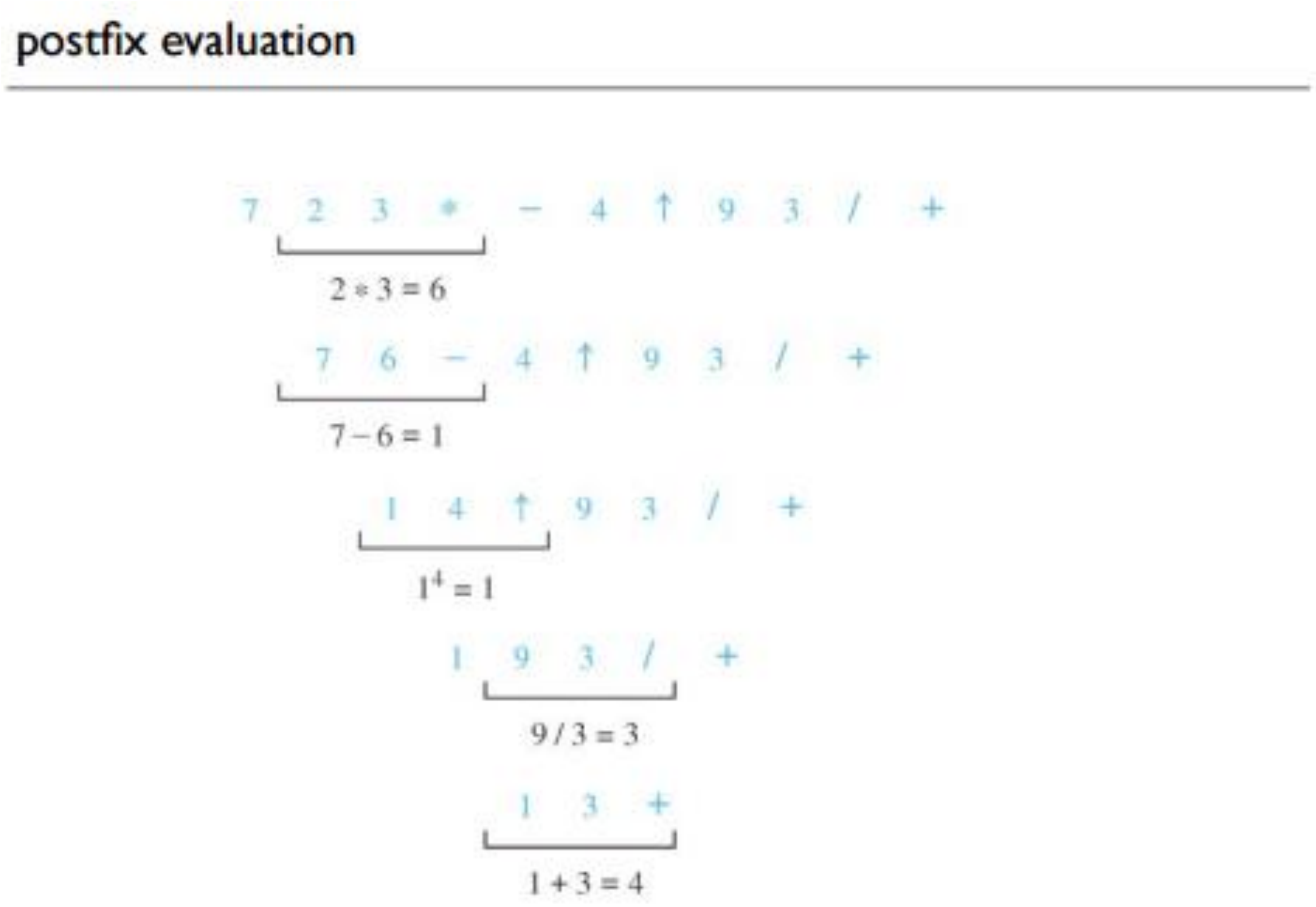
Category	Evaluation	
p4_1	50	
p4_2	50	
Total	100	

- *Use GCC 4.8 version or GCC 5.4 version.*
- *No score will be given if the gcc version is different.*

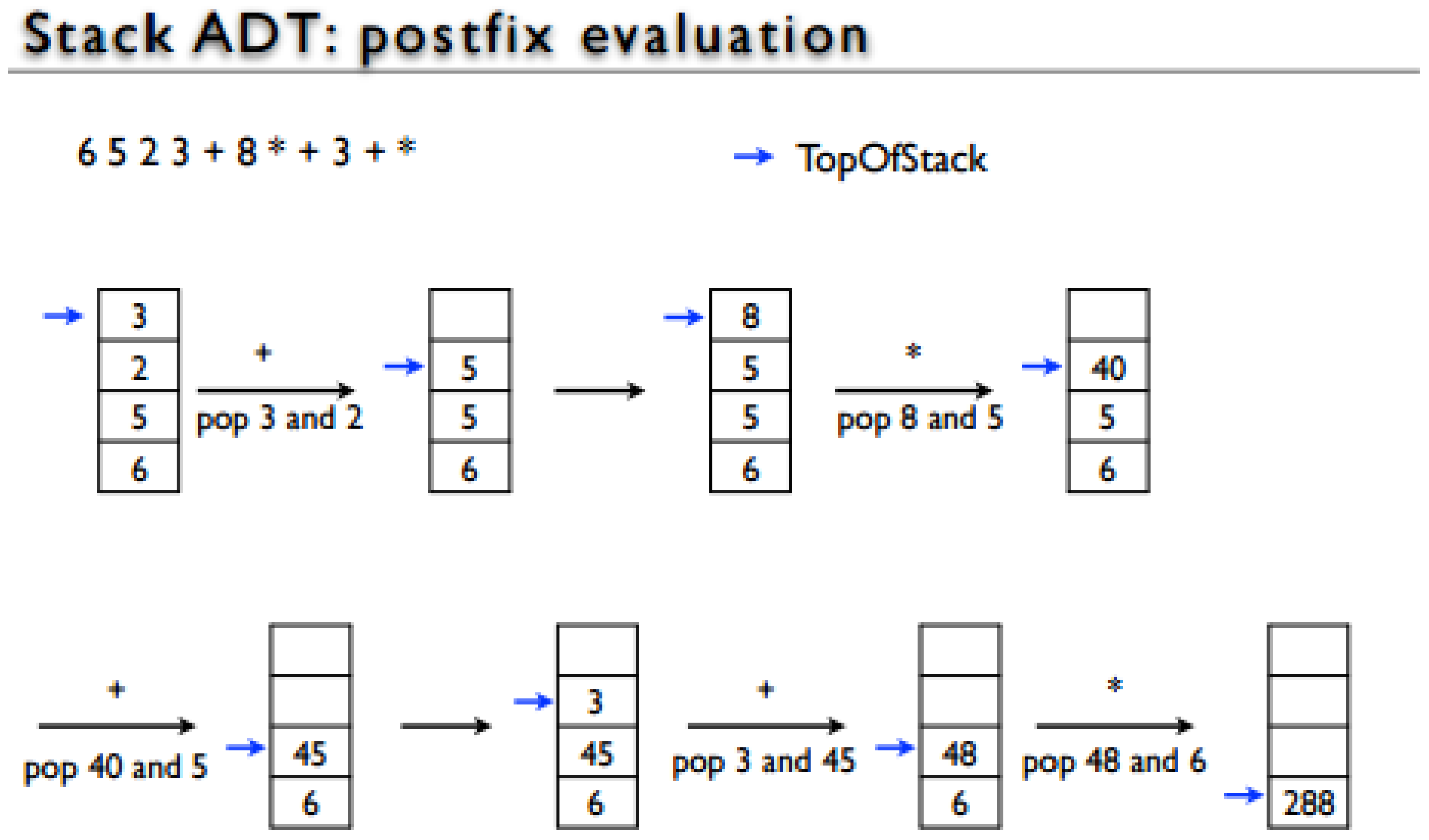
Lab4

- You should finish p4_1 (Push, CreateStack, IsFull) during the lab session and submit it to git before you leave.
- For p4_2 (Pop, DeleteStack , IsEmpty) you can submit it to the git later.
- Folder name : Lab4
- code name: p4_1, p4_2
- -15 score , if the folder, code names are wrong.
- -5 per code, if it does not use FILE I/O
- Each code will be tested by 5 different input files.
- 10 score for each input, if you don't get the answer you get 0 score.

Lab4 – postfix evaluation



Lab4 – postfix evaluation using Stack



Lab4 – Stack ADT

- 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 '#'.
- 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 and perform the operation, and push the result back to the stack.

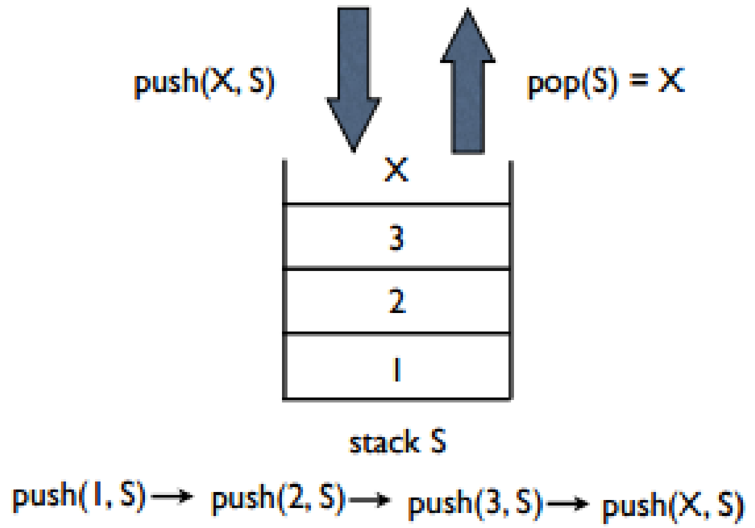
Lab4 – Stack ADT

- **CreateStack** create a new stack with the size of max.
- **Push** push a new element at the end of the element in the stack. If you stack is full, just print an error message.
- **Pop** pop the element in the end of the stack. If stack does not have any element, just print an error message.
- **DeleteStack** free all the memory allocated to stack.
- **IsFull** check if the stack is full.
- **IsEmpty** check if the stack is empty.

Lab4 –Stack ADT

- Structure

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



Lab4 – Stack ADT

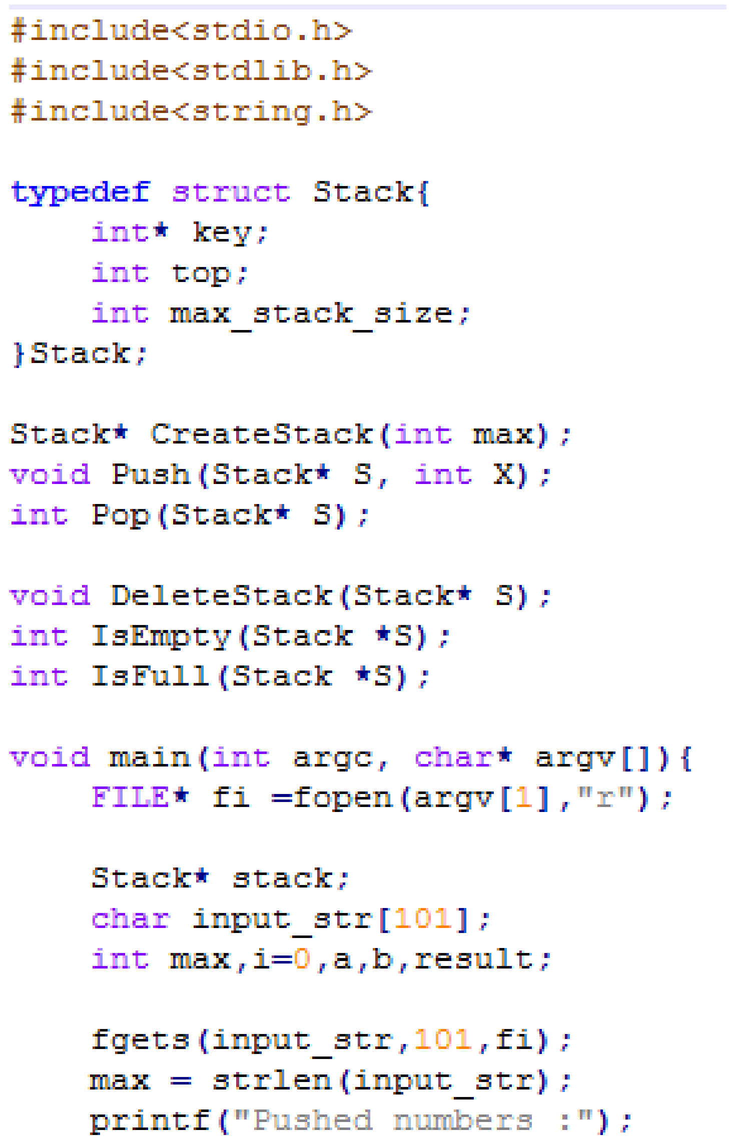
- Structure

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

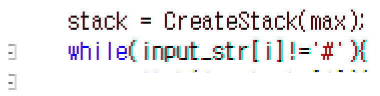
- Function

```
<Lab4>
Stack* CreateStack(int max);
void Push(Stack* S, int X);
int IsFull(Stack *S);
<HW4>
int IsEmpty(Stack *S);
int Pop(Stack* S);
void DeleteStack(Stack* S);
```

Lab 4 Stack ADT - *Push*

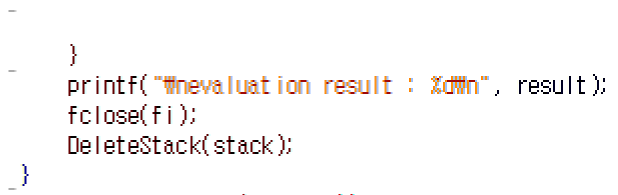


```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

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);

void DeleteStack(Stack* S);
int IsEmpty(Stack *S);
int IsFull(Stack *S);

void main(int argc, char* argv[]){
    FILE* fi =fopen(argv[1],"r");

    Stack* stack;
    char input_str[101];
    int max,i=0,a,b,result;

    fgets(input_str,101,fi);
    max = strlen(input_str);
    printf("Pushed numbers :");
```



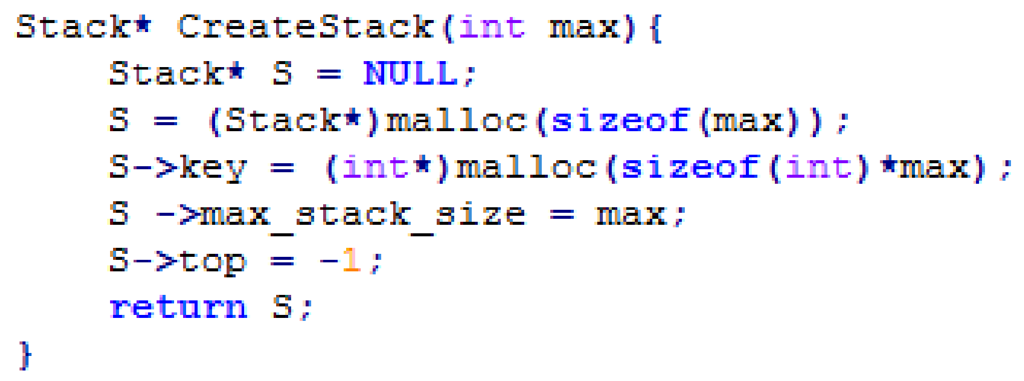
```
    stack = CreateStack(max);
    while(input_str[i]!='#'){
```

//Push(S, input_str[i])
//Pop(S)



```
    }
    printf("₩nevaluation result : %d₩n", result);
    fclose(fi);
    DeleteStack(stack);
}
```

Lab 4. Stack ADT - Push

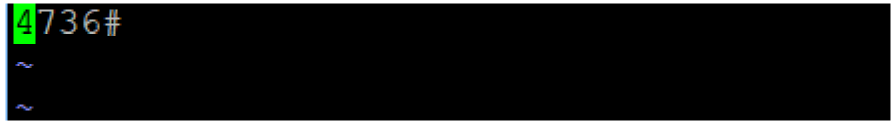


```
Stack* CreateStack(int max){
    Stack* S = NULL;
    S = (Stack*)malloc(sizeof(max));
    S->key = (int*)malloc(sizeof(int)*max);
    S ->max_stack_size = max;
    S->top = -1;
    return S;
}
```

```
void Push(Stack* S, int X)
{






            printf(“%d  inserted₩n”, X);

}
```

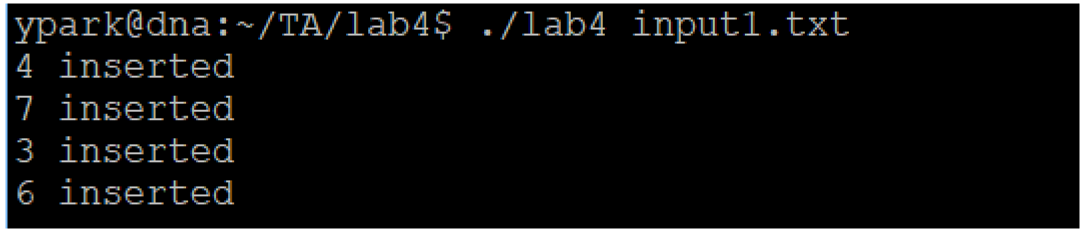
Lab 4. Postfix Evalution using Stack

- input file : lab4_input1.txt



```
4736#
~
~
```

- Result

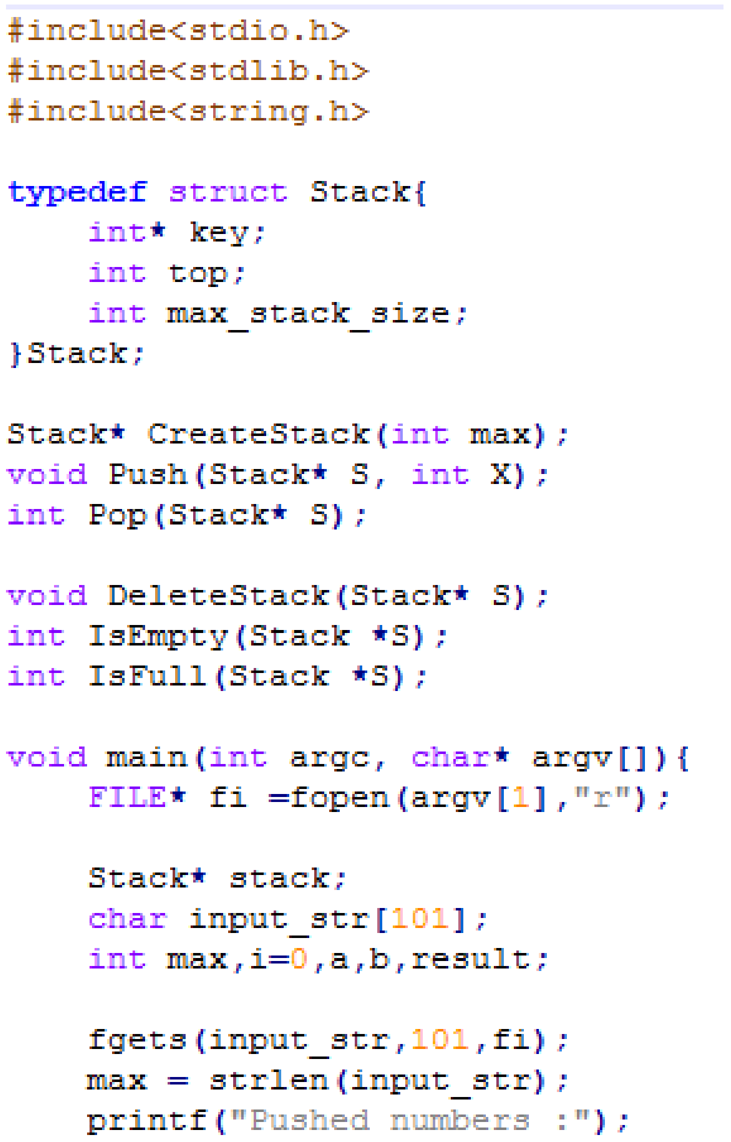


```
ypark@dna:~/TA/lab4$ ./lab4 input1.txt
4 inserted
7 inserted
3 inserted
6 inserted
```

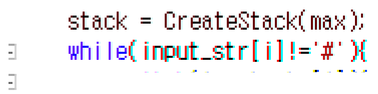
Lab 4. Stack ADT - Push

- program name : p4_1.c
- input : a list of numbers in a file.
- output : the corresponding result in the standard output.

Lab 4 Stack ADT - Push

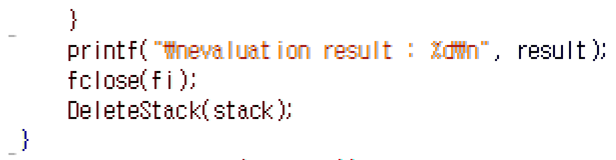


```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

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);

void DeleteStack(Stack* S);
int IsEmpty(Stack *S);
int IsFull(Stack *S);

void main(int argc, char* argv[]){
    FILE* fi =fopen(argv[1],"r");

    Stack* stack;
    char input_str[101];
    int max,i=0,a,b,result;

    fgets(input_str,101,fi);
    max = strlen(input_str);
    printf("Pushed numbers :");
```



```
    stack = CreateStack(max);
    while( input_str[i]!='#' ){
```

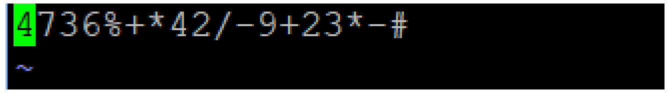
//do operation



```
    }
    printf("₩nevaluation result : %d₩n", result);
    fclose(fi);
    DeleteStack(stack);
}
```

Lab 4. Postfix Evalution using Stack

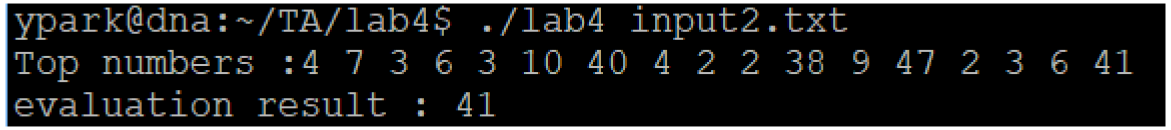
- input file : lab4_input2.txt



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

- Result

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



```
ypark@dna:~/TA/lab4$ ./lab4 input2.txt
Top numbers :4 7 3 6 3 10 40 4 2 2 38 9 47 2 3 6 41
evaluation result : 41
```

HW 4. Postfix evaluation using Stack

- program name : p4_2.c
- input : a list of operations in a file.
- output : the corresponding result in the standard output.