<push 함수>

```
void push(int value, const char* info)
{
    if (SP >= STACK_SIZE - 1) {
        printf( format: "ERROR : Stack Overflow\n");
        return;
    }
    SP++;
    call_stack[SP] = value;
    snprintf(stack_info[SP], n: sizeof(stack_info[SP]), format: "%s", info);
}
```

스택이 가득 찬 경우, Stack Overflow 오류 메시지를 출력한다.

푸쉬가 들어오는 경우에는 스택 포인터를 증가시켜 새로운 항목을 추가할 공간을 만든다.

Push 함수의 인자로 value를 받아, 스택의 SP번째 index에 해당하는 공간에 데이터를 저장한다.

스택의 이름을 stack_info 배열에 저장한다.

<pop 함수>

```
int pop()
{
    if (SP == -1) {
        printf( format: "ERROR : Stack Underflow\n");
        return -1;
    }
    int value = call_stack[SP];
    SP--;
    return value;
}
```

스택이 비어있으면, 즉 SP가 -1이면 Stack Underflow 오류 메시지를 출력한다.

Pop 명령은 가장 최근의 값을 꺼내므로, 스택에서 가장 최근의 값인 call_stack[SP]를 꺼낸다.

스택 포인터를 1 감소시켜 이전 값을 가리키게 하고, 꺼내기로 했던 값을 그대로 return해준다.

<prologue 함수>

```
void prologue()
{
    push(FP, info: "old EBP");
    FP = SP;
}
```

현재 FP(프레임 포인터)의 값을 스택에 저장한다.

새로운 FP 값을 SP 값으로 지정한다.

<epilogue 함수>

```
void epilogue()
{
    SP = FP;
    FP = pop();
}
```

FP를 SP에 할당해서 스택을 복원한다.

Pop 함수로 프레임 포인터를 복원한다.

<흐름>

main에서 func1이 호출된다.

```
====== Current Call Stack ======
4 : arg3 = 3    <=== [esp]
3 : arg2 = 2
2 : arg1 = 1
1 : var_1 = 100
0 : old EBP    <=== [ebp]
================================
```

Prologue가 실행되며 새로운 스택 프레임이 생성되고, 주어진 var_1=100, arg1=1, arg2=2, arg3=3 이 차례로 스택이 푸시된다. 이때의 스택 상태는 위와 같다.

```
====== Current Call Stack ======
8 : arg2 = 13    <=== [esp]
7 : arg1 = 11
6 : var_2 = 200
5 : old EBP = 0    <=== [ebp]
4 : arg3 = 3
3 : arg2 = 2
2 : arg1 = 1
1 : var_1 = 100
0 : old EBP
================================
```

func1 함수 내에서 func2가 호출된다. func2 함수가 호출될 때도 Prologue가 실행되며 새로운 스택 프레임이 생성되고, 주어진 var_2=200, arg1=11, arg2=13이 스택에 푸시된다. 이때의 스택 상태는 위와 같다. ebp는 func1-fun2 사이인 5번에 걸려있으며, old EBP를 func1 전의 0으로 걸어놓고 있다. esp는 가장 위인 8번에 걸려있다.

```
====== Current Call Stack ======
12 : arg1 = 77    <=== [esp]
11 : var_4 = 400
10 : var_3 = 300
9 : old EBP = 5    <=== [ebp]
8 : arg2 = 13
7 : arg1 = 11
6 : var_2 = 200
5 : old EBP = 0
4 : arg3 = 3
3 : arg2 = 2
2 : arg1 = 1
1 : var_1 = 100
0 : old EBP
================================
```

func2 함수 내에서 func3이 호출된다. func3 함수가 호출될 때도 Prologue가 실행되며 새로운 스택 프레임이 생성되고, 주어진 var_3=300, var_4=400, arg1=77이 스택에 푸시된다. 이때의 스택 상태는 위와 같다. ebp는 func2-fun3 사이인 9번에 걸려있으며, old EBP를 func1-fun2 사이인 5로 걸어놓고 있다. esp는 가장 위인 12번에 걸려있다.

```
====== Current Call Stack ======
8 : arg2 = 13    <=== [esp]
7 : arg1 = 11
6 : var_2 = 200
5 : old EBP = 0    <=== [ebp]
4 : arg3 = 3
3 : arg2 = 2
2 : arg1 = 1
1 : var_1 = 100
0 : old EBP
================================
```

func3이 종료된 이후, epilogue가 실행되며 SP와 FP가 복원되며 func3에서 들어왔던 값들이 pop 된다. ebp는 원래의 ebp인 9번에서 걸어놓았던 5번으로 복원되었고, esp도 현재 가장 위인 8번이 되었다.

```
====== Current Call Stack ======
4 : arg3 = 3    <=== [esp]
3 : arg2 = 2
2 : arg1 = 1
1 : var_1 = 100
0 : old EBP    <=== [ebp]
================================

Stack is empty.
```

func2가 종료된 이후, epilogue가 실행되며 SP와 FP가 복원되며 func2에서 들어왔던 값들이 pop 된다. ebp는 원래의 ebp인 5번에서 걸어놓았던 0번으로 복원되었고, esp도 현재 가장 위인 4번이 되었다.

func1에서 남은 모든 값들을 pop하자, Stack에 남아있는 값이 더 없게 되어 "Stack is empty."라는 문구가 출력된다.