

CSED101. Programming & Problem solving

Spring, 2019

Programming Assignment #2 (60 points)

이지형 (sg01257@postech.ac.kr)

■ **Due:** 2019.04.04

■ **Development Environment:** Windows Visual Studio 2017

■ **제출물**

- **C Code files (*.c)**
 - 확장자를 포함한 소스파일의 이름은 assn2.c로 할 것.
 - 파일명 양식을 미 준수 하거나 프로젝트 폴더를 통째로 제출 할 시 감점
 - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 주석을 붙일 것.
- **보고서 파일** (.doc(x) or .hwp) 예) assn2.doc(x) 또는 assn2.hwp
 - AssnReadMe.pdf 를 참조하여 작성할 것.
 - 프로그램 실행화면을 캡처하여 보고서에 포함시키고, 간단히 설명할 것 !!
 - 명예서약(Honor code): 표지에 다음의 내용을 포함한다. "나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다." 보고서 표지에 명예서약이 없는 경우는 과제를 제출하지 않은 것으로 처리한다.
 - 소스코드와 보고서 파일을 LMS를 이용하여 제출한다.

■ **주의사항**

- 각 문제에 해당하는 요구사항을 반드시 지킬 것.
- 모든 문제의 출력 형식은 아래의 예시들과 동일해야 하며, 같지 않을 시는 감점이 된다.
- **컴파일 & 실행이 안되면 무조건 0점 처리된다.**
- 하루 late시 20%가 감점되며, 3일 이상 지나면 받지 않는다. (0점 처리)
- 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 'POSTECH 전자컴퓨터공학부 부정행위 정의'를 따른다. (LMS의 과목 공지사항의 제목 [document about cheating]의 첨부파일인 disciplinary.pdf를 참조할 것.)
- 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.

■ Problem: 하노이의 탑 퍼즐 구현

(목적)

이번 과제를 통하여 조건문, 반복문, 사용자 정의 함수 사용법을 익힌다.

(주의사항)

- 이번 과제는 함수를 정의하고 사용하는 방법을 익히는 문제이므로 사용자 정의 함수를 사용하지 않고, main함수에 모든 기능을 구현한 경우 감점 처리 함. 아래 설명에서 반드시 정의해서 사용해야 할 사용자 정의 함수가 설명되어 있으니 확인한 후 구현한다. 그 외의 필요한 함수를 정의해서 사용할 수 있다.
- 이 때, 설명에서 지정한 사용자 정의 함수가 받는 인자들의 개수와 자료형은 변경 가능하다. 단, 함수의 이름과 기능은 설명대로 동일하게 구현한다.
- 전역 변수, 배열 및 goto 문은 사용할 수 없다.
- 사용자로부터 입력 받을 때, 정수 외의 입력에 대해서는 고려하지 않아도 된다.
- 문제의 출력 형식은 실행 예시와 최대한 비슷하게 작성해 주세요.

(설명)

하노이의 탑(https://en.wikipedia.org/wiki/Tower_of_Hanoi)의 규칙을 구현하고 블록을 옮길 수 있게 하는 프로그램을 작성한다.

먼저 1부터 5 사이의 숫자를 입력 받아 탑의 층 수를 결정하고, 입력을 통해 각 기둥에 있는 블록이 다른 기둥으로 옮겨질 수 있도록 한다. 첫번째 기둥의 모든 블록이 규칙에 알맞게 두번째나 세 번째 기둥으로 옮겨질 경우 프로그램을 종료한다.

- 하노이의 탑 규칙은 다음과 같다.
 - (1) 한번에 하나의 블록만 이동할 수 있다.
 - (2) 블록을 이동할 경우, 최 상단의 블록 하나만 다른 기둥으로 옮길 수 있다.
 - (3) 작은 블록 위에는 큰 블록이 올 수 없다.

1. 탑의 층 수 결정

프로그램이 시작되면, 그림 1처럼 1부터 5 사이의 정수로서 탑의 층 수를 입력 받도록 한다.

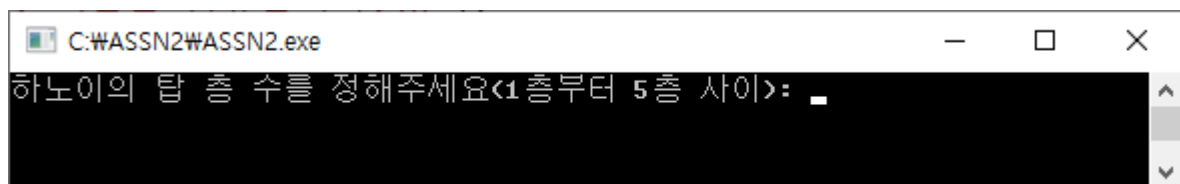


그림 1. 탑 층수 선택

이 때의 입력은 숫자만 입력된다고 가정하고, 숫자 외의 다른 입력이 들어오는 경우는 고려하지 않는다. 단, 1부터 5까지의 숫자가 아닌 -1, 0, 6, 999등의 **범위 외의 입력의 경우**, 그림 2와 같이 오류 메시지를 띄우고 제대로 된 값이 들어올 때까지 다시 층수를 입력 받는다.

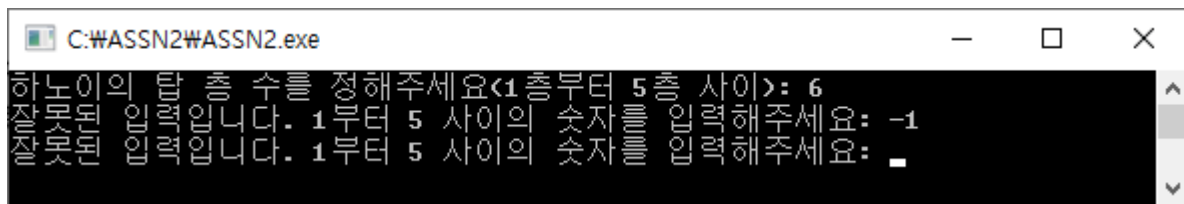


그림 2. 잘못된 범위의 숫자가 입력된 경우

2. 탑 출력

그림 1에서 제대로 된 입력이 들어왔을 경우, 구분선('= ' 40개 두 줄)을 출력하며, 또 '>>>Step 1'을 출력 후, 그림 3과 같이 첫번째 기둥에 입력된 층수에 해당하는 탑을 출력한다.

※ 'Step 1' 출력 부분은 '4. Step 증가 및 퍼즐 종료 결정'을 참고하여 구현한다.

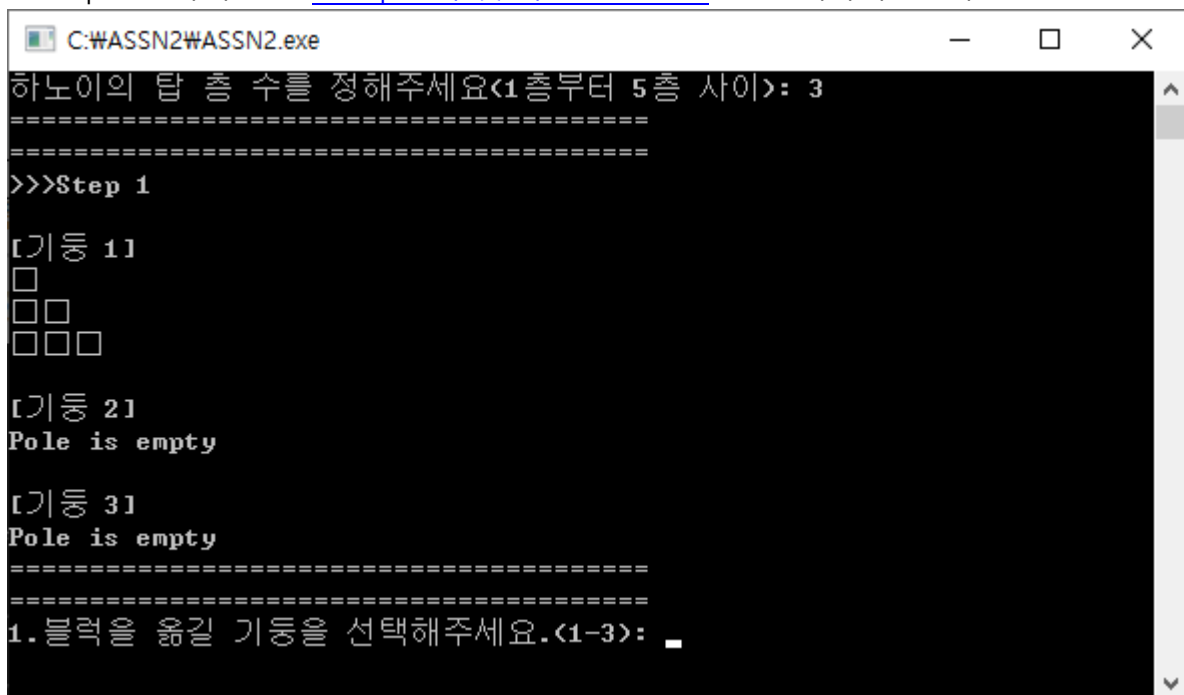


그림 3. 각 기둥의 출력

첫번째 탑(기둥 1)에는 3층일 경우 3개, 2개, 1개의 블록이 차례대로 쌓여있고, 5층일 경우 5개, 4개, ..., 1개의 블록이 차례대로 쌓여있다. 기둥 2와 기둥 3의 경우 Step 1에선 비어 있으므로, 'Pole is empty' 문자열을 대신 출력한다.

기둥의 출력이 끝나면, 다시 구분선을 출력하고, 블록을 옮길 기둥을 입력 받게 한다.

➤ 이 때, 반드시 아래의 사용자 정의 함수를 정의 한 후 사용한다.

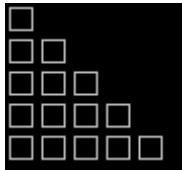
- **print_divider()**: 구분선 출력을 위한 함수로 '= ' 40개 두 줄을 출력한다.

이때 =는 **for문을 이용해서 출력한다.** (printf("===...===");와 같은 식으로 처리하는 경우 감점)

- **print_poles():** 각 기둥에 해당되는 정보를 매개변수로 전달받아서, 기둥 1, 2, 3에 해당되는 탑을 전부 출력한다. 아래의 **print_one_pole()**함수를 호출하여 각각의 탑을 출력하도록 한다.
※ '□' 문자 입력: 'ㅁ'키 입력 후, 한자키를 눌러 해당 문자를 선택함으로써 할 수 있다.
□ 대신 다른 네모형 문자를 사용해도 상관없다.

- **print_one_pole():** 기둥 한 개의 정보를 전달받아, 아래와 같이 탑을 출력한다. (반드시 2중 반복문을 사용하여 구현 할 것)

예시)



(구현 힌트) 마지막 장의 **Tips**를 참고한다.

3. 블록 이동

기둥 출력이 끝나면, 그림 4와 같은 문구와 함께 블록을 옮길 기둥을 입력 받는다.

```

C:\WASSN2\WASSN2.exe
=====
>>>Step 1
[기둥 1]
□
□□
□□□

[기둥 2]
Pole is empty

[기둥 3]
Pole is empty
=====
=====
1.블록을 옮길 기둥을 선택해주세요.<1-3>: 1
2.블록이 옮겨질 기둥을 선택해주세요.<1-3>:

```

그림 4. 기둥 선택

알맞은 기둥을 선택한 경우, 그 다음으로 블록이 옮겨질 기둥을 입력 받게 된다.

앞서 선택한 기둥과 같은 기둥을 선택한 경우, 블록을 이동하지 않은 채 그대로 두고 [4. Step 증가 및 퍼즐 종료 결정](#)으로 넘어간다.

➤ 블록을 옮길 기둥 선택 시, 다음과 같이 2가지 예외처리를 하도록 한다.

- 1) 사용자가 비어있는 기둥을 선택 할 시, 그림 5와 같이 경고 메시지와 함께 제대로 된 입력이 들어올 때까지 재 입력을 받는다.
- 2) 1부터 3 사이의 숫자가 아닌 다른 숫자를 입력할 경우, 그림 6처럼 경고 메시지와 함께 제대로 된 입력이 들어올 때까지 재 입력을 받는다.

```

C:\WASSN2\WASSN2.exe
하노이의 탑 층 수를 정해주세요<1층부터 5층 사이>: 3
=====
>>>Step 1

[기둥 1]
□
□□
□□□

[기둥 2]
Pole is empty

[기둥 3]
Pole is empty
=====
1. 블록을 옮길 기둥을 선택해주세요.<1-3>: 2
1. 해당 기둥이 비어있습니다. 비어있지 않은 기둥을 선택해주세요: _

```

그림 5. 블록을 옮길 기둥 선택에서 비어있는 기둥을 선택

```

C:\WASSN2\WASSN2.exe
하노이의 탑 층 수를 정해주세요<1층부터 5층 사이>: 3
=====
>>>Step 1

[기둥 1]
□
□□
□□□

[기둥 2]
Pole is empty

[기둥 3]
Pole is empty
=====
1. 블록을 옮길 기둥을 선택해주세요.<1-3>: -1
1. 잘못된 입력입니다. 1부터 3 사이의 숫자를 입력해주세요: 5
1. 잘못된 입력입니다. 1부터 3 사이의 숫자를 입력해주세요: _

```

그림 6. 블록을 옮길 기둥 선택에서 숫자 범위 입력 오류

- 블록이 옮겨질 기둥 선택 시, 다음 2가지 예외처리를 하도록 한다.
 - 1) 1부터 3 사이의 숫자가 아닌 다른 숫자를 입력할 경우, 그림 7처럼 경고 메시지와 함께 제대로 된 입력이 들어올 때까지 재 입력을 받는다.
 - 2) 선택된 기둥으로 블록을 옮기려고 할 때, 하노이 탑 규칙에 어긋난 기둥이 선택된 경우, 그림 8처럼 경고 메시지와 함께 제대로 된 입력이 들어올 때까지 재 입력을 받는다.
- 해당 예외처리를 위해서, 반드시 아래의 사용자 정의 함수를 정의 한 후 사용한다.
 - **check_valid_move():** 기둥 정보들과 선택된 기둥 번호들을 매개변수로 전달받아서, 하노이의 탑 규칙에 맞으면 1을 리턴하고, 그렇지 않을 경우 0을 리턴한다.
 - **check_not_empty():** 기둥 정보들과 선택된 기둥 번호를 전달받아서, 해당 기둥에 블록이 있으면 1을, 그렇지 않을 경우 0을 리턴한다.

```

C:\WASSN2\WASSN2.exe
하노이의 탑 층 수를 정해주세요<1층부터 5층 사이>: 3
=====
>>>Step 1

[기둥 1]
□
□□
□□□

[기둥 2]
Pole is empty

[기둥 3]
Pole is empty
=====
=====
1.블록을 옮길 기둥을 선택해주세요.<1-3>: 1
2.블록이 옮겨질 기둥을 선택해주세요.<1-3>: 4
2.잘못된 입력입니다. 1부터 3 사이의 숫자를 입력해주세요: _
  
```

그림 7. 블록이 옮겨질 기둥 선택에서 숫자 범위 입력 오류

```
C:\WASSN2\WASSN2.exe
Block Moved!
=====
>>>Step 2

[기둥 1]
□□
□□□

[기둥 2]
□

[기둥 3]
Pole is empty
=====
=====
1.블럭을 옮길 기둥을 선택해주세요.<1-3>: 1
2.블럭이 옮겨질 기둥을 선택해주세요.<1-3>: 2
2.하노이의 탑 규칙에 어긋납니다. 다른 기둥을 선택해주세요: _
```

그림 8. 옮겨지는 블록이 하노이의 탑 규칙에 어긋날 시

예외 사항이 발생하지 않고, 그림 9처럼 제대로 숫자가 입력된 상태에서 Enter를 누를 시 다음 스텝으로 넘어간다. 위에서 서술하지 않은 다른 예외에 대해선 고려하지 않는다.

```
C:\WASSN2\WASSN2.exe
하노이의 탑 층 수를 정해주세요<1층부터 5층 사이>: 3
=====
>>>Step 1

[기둥 1]
□
□□
□□□

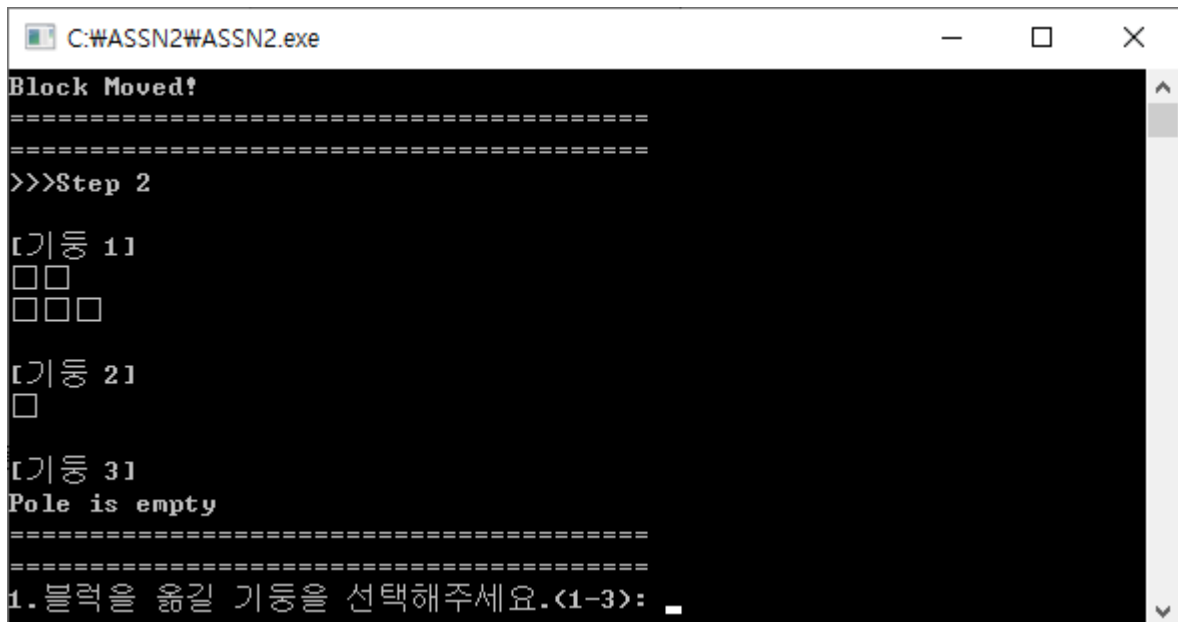
[기둥 2]
Pole is empty

[기둥 3]
Pole is empty
=====
=====
1.블럭을 옮길 기둥을 선택해주세요.<1-3>: 1
2.블럭이 옮겨질 기둥을 선택해주세요.<1-3>: 2_
```

그림 9. 예외 미발생 케이스

4. Step 증가 및 퍼즐 종료 결정

블록을 주고받을 기둥들이 결정되면, 그림 10처럼 화면을 지우고 Block Moved! 문구를 출력한 뒤, Step을 증가시킨다. (화면을 지우는 방법은 아래 Tips.을 참고하여 구현한다.)



```
C:\WASSN2\WASSN2.exe
Block Moved!
=====
>>>Step 2

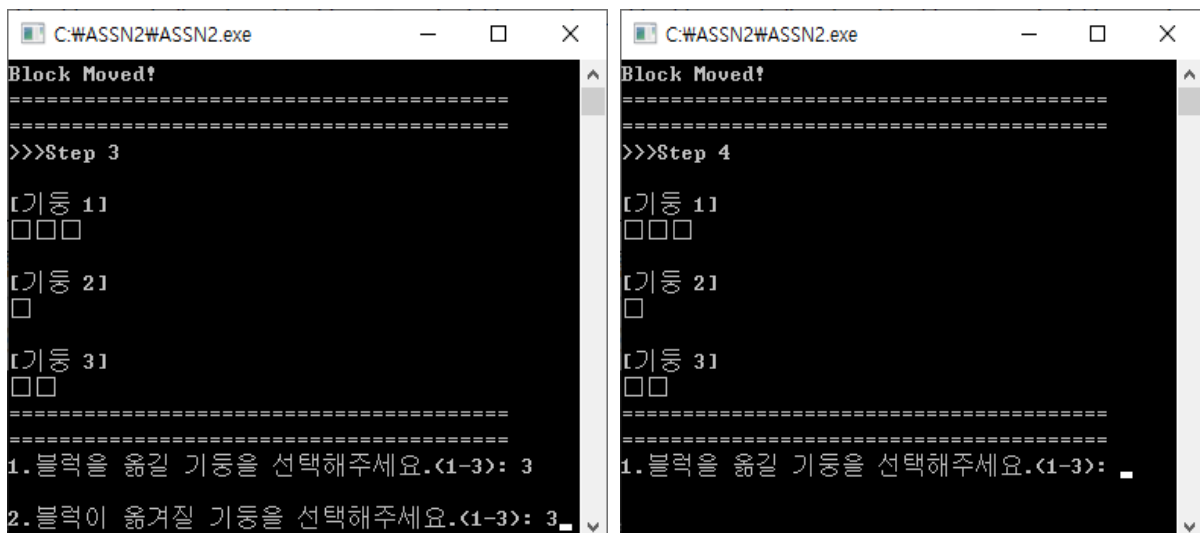
[기둥 1]
□□
□□□

[기둥 2]
□

[기둥 3]
Pole is empty
=====
1. 블록을 옮길 기둥을 선택해주세요.<1-3>: _
```

그림 10. 블록이 [기둥 1]에서 [기둥 2]로 이동된 후의 화면

블록이 옮겨지는 기둥과 블록을 받는 기둥이 똑같은 경우에도 그림 11과 같이 Block Moved! 문구를 출력하고 Step을 증가시킨다.



```
C:\WASSN2\WASSN2.exe
Block Moved!
=====
>>>Step 3

[기둥 1]
□□□
[기둥 2]
□
[기둥 3]
□□
=====
1. 블록을 옮길 기둥을 선택해주세요.<1-3>: 3
2. 블록이 옮겨질 기둥을 선택해주세요.<1-3>: 3_

C:\WASSN2\WASSN2.exe
Block Moved!
=====
>>>Step 4

[기둥 1]
□□□
[기둥 2]
□
[기둥 3]
□□
=====
1. 블록을 옮길 기둥을 선택해주세요.<1-3>: _
```

그림 11. 블록이 [기둥 3]에서 [기둥 3]으로 이동된 후의 화면

Step은 선택이 정상적으로 끝날 때마다 증가 되어야 하며, 이동된 블록들의 모습들 또한 기둥들을 출력하는 함수인 `print_poles()`로 출력될 수 있도록 한다.

위와 같은 방법을 반복해, 모든 블록이 처음 [기둥 1]에 쌓여져 있던 모습 그대로 다른 기둥으로 이동된 경우 그림 12과 같이 출력 후 프로그램을 종료한다.

- 이 때, 종료 여부를 판단하기 위해 반드시 아래의 사용자 정의 함수를 정의 한 후 사용한다.
- **check_finish()**: 현재 기둥들의 정보가 종료 조건과 일치할 시 1을, 아니면 0을 리턴한다.

```

Microsoft Visual Studio 디버그 콘솔
Block Moved!
=====
=====
>>>Step 8

[기둥 1]
Pole is empty

[기둥 2]
Pole is empty

[기둥 3]
□
□□
□□□
=====
=====
블럭이 모두 다른 기둥으로 옮겨졌습니다. 퍼즐을 종료합니다.

c:\Users\j\i\hyung\source\repos\ASSN2\Debug\ASSN2.exe(9060 프로세스)
이(가) 0 코드로 인해 종료되었습니다.
디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구]->[옵션]->[디버
깅]->[디버깅이 중지되면 자동으로 콘솔 닫기]를 사용하도록 설정합니
다.
이 창을 닫으려면 아무 키나 누르세요.

```

그림 12. 퍼즐 종료

Tips.

- **print_poles() 관련 팁**

인자를 전달할 때, int형 parameter 세 개로 모든 정보를 넘길 수 있는 방법을 생각해보자. 예를 들어, 1,2,5개의 블록이 차례대로 쌓여있는 기둥의 경우, 숫자 125를 전달해 자릿수로 끊어서 출력을 하는 방법이 있다.

- **화면 지우기**

헤더파일 <stdlib.h>를 include한 뒤, system 함수에 매개변수로 cls를 전달하면, 앞서 출력한 화면이 지워진다.

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

int main()
{
    printf("Something\n");
    system("cls");
    printf("Something2\n");
    return 0;
}
```

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

int main()
{
    printf("Something\n");
    //system("cls");
    printf("Something2\n");
    return 0;
}
```

위 두 코드를 각각 실행해봄으로써 차이를 파악해보자.

※ 리눅스에서 테스트 해보길 원하는 경우, system("cls"); 대신에 system("clear");로 변경하면 된다.