CSED101. Programming & Problem solving
Fall 2023

Programming Assignment #1
(75 points)

김인호 (kimih@postech.ac.kr)

■ **제출 마감일: 2023.10.17 23:59**
■ **파이썬 버전: Python 3.x**

■ **제출물**

- .py 소스 코드 (assn1.py)
  - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 **주석**을 붙일 것.
- 보고서 파일 (.docx, .hwp 또는 .pdf; assn1.docx, assn1.hwp 또는 assn1.pdf)
  - 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 PLMS를 통하여 제출한다.
  - PLMS에 제출할 때는 소스파일과 보고서를 압축하지 말고 각각 업로드 할 것.

■ **주의사항**

- 구문 오류(Syntax Error)가 발생하거나 실행이 되지 않는 과제는 0점으로 채점됩니다.
- 제출 기한보다 하루 늦게 제출된 과제는 최종 20%, 이틀 늦게 제출된 과제는 최종 40% 감점됩니다. 제출 기한보다 사흘 이상 늦으면 제출 받지 않습니다 (0점 처리).
  늦은 제출시 PLMS에 기록된 최종 수정일시를 기준으로 감점합니다.
- 각 문제의 제한 조건과 요구 사항을 반드시 지켜 주시기 바랍니다.
- 모든 문제의 출력 형식은 채점을 위해 아래에 제시된 예시들과 최대한 비슷하게 작성해 주세요.
- 각 문제에 명시된 에러 처리 외에는 고려하지 않아도 됩니다.
- 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.
- 운영체제 및 개발 환경마다 화면 지우기 명령어가 달라지므로 화면 지우기 요구사항을 반드시 따라야 합니다.
- 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.)

■ Problem: 묵찌빠 계단오르기

(목적)

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

(주의사항)

(1) 이번 과제는 함수를 정의하고 사용하는 방법을 익히는 문제이므로 **함수를 정의하지 않고 기능을 구현한 경우 감점 처리된다.** 문서에 반드시 정의해서 사용해야 할 사용자 정의 함수가 설명되어 있으니 확인 후 구현하도록 한다.

- 정의해서 사용해야 할 사용자 정의 함수의 매개변수는 자유롭게 변형하여 사용할 수 있다.

(2) int, float, bool, str, list, tuple 자료형까지 수업시간에 다룬 내용에 한해서 사용 가능하다.

(3) **문제의 출력 형식은 채점을 위해 아래의 실행 예시와 최대한 비슷하게 작성해야 한다**.

(설명)

이 게임은 2인으로 진행이 되며, 본 과제에서는 컴퓨터와 사용자가 플레이를 하게 된다.
컴퓨터와 사용자가 서로 맞은편 언덕 위 계단에서 묵찌빠를 하면서 규칙에 따라 계단을 오르거나 내려가게 된다. 이때, 먼저 상대편의 시작점에 도착하는 플레이어가 이기는 게임이다.

승패를 위한 '묵찌빠'는 아래 규칙을 사용하도록 한다.

(1) 공격권 결정 가위바위보

게임을 위해 각 플레이어(컴퓨터, 사용자)들은 가위, 바위, 보 중에 하나를 선택하여 낸다. 만약 두 플레이어가 같은 선택을 한 경우 승부가 날 때까지 가위바위보를 계속하며 가위바위보 승부에서 이긴 플레이어가 공격권을 갖는다.

(2) 묵찌빠 수행

플레이어들은 가위, 바위, 보 중 하나를 선택한다. 공격권을 가진 사람의 손 모양이 상대의 손 모양과 일치하면 공격권을 가진 사람이 승리한다. 승부가 갈리지 않은 경우 현 상태에서 가위바위보 규칙 상 이긴 사람이 공격권을 가져간다.

(3) 승패 결정 및 계단 이동 규칙

묵찌빠에서 승리한 플레이어가 승부가 날 때까지 소모된 턴 수만큼 계단을 이동한다.

I. 게임 시작

프로그램이 시작되면, 아래 그림 1과 같이 계단의 개수를 입력 받기 위한 텍스트가 출력되고 사용자로부터 입력을 받을 준비를 한다. 단 계단의 개수는 **10개~30개로 한정**한다. 범위를 벗어난 입력에 대해서는 아래의 예시처럼 다시 입력 받도록 한다. **정수 이외의 입력은 고려하지 않는다.**

(실행 예시의 빨간색 밑줄은 사용자 입력에 해당한다.)

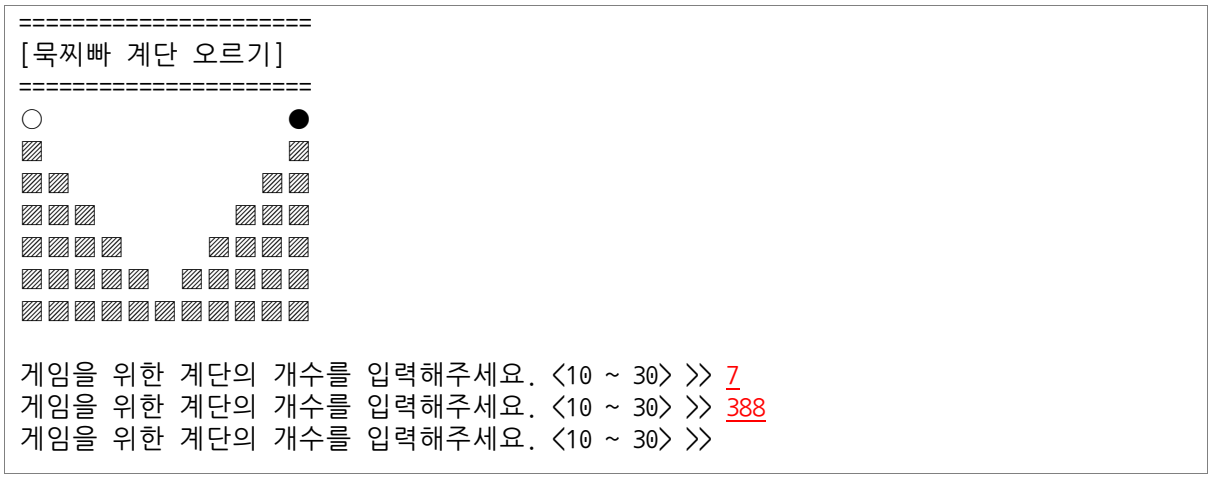


그림 1. 게임 초기 화면

범위 내의 계단의 개수를 입력하면 화면을 지운 후 컴퓨터와 플레이를 할 화면이 그림 2와 같이 출력되도록 한다. 이 부분 구현에 대해서는 Tips(마지막 쪽)의 '화면 지우기' 설명을 참조한다. (이 과제의 모든 출력화면은 화면을 지운다고 명시한 경우, 그에 맞게 지운 후 출력한다. 화면 지우기를 수행할 시 엔터 이외의 입력은 고려하지 않는다.)

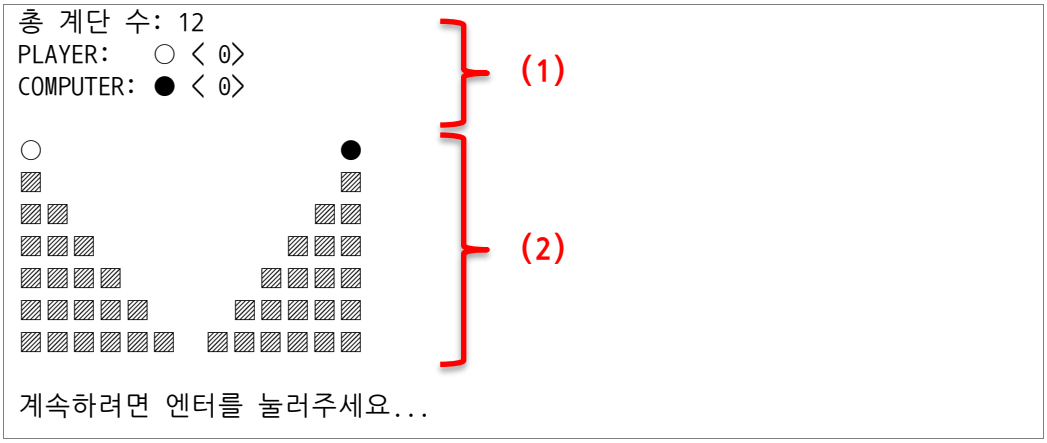


그림 2. 계단이 생성된 화면 예시(12개의 계단을 입력한 경우)

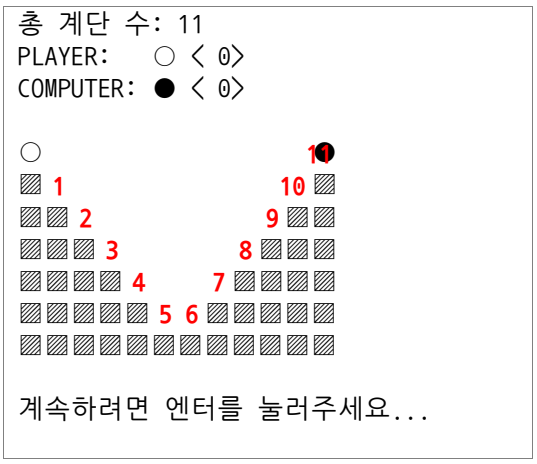
◆ **그림 2의 (1)**

- 사용자가 입력한 총 계단 수
- PLAYER의 위치를 표시하는 기호 '○'와 현재까지 이동한 계단 수
- COMPUTER의 위치를 표시하는 기호 '●'와 현재까지 이동한 계단 수

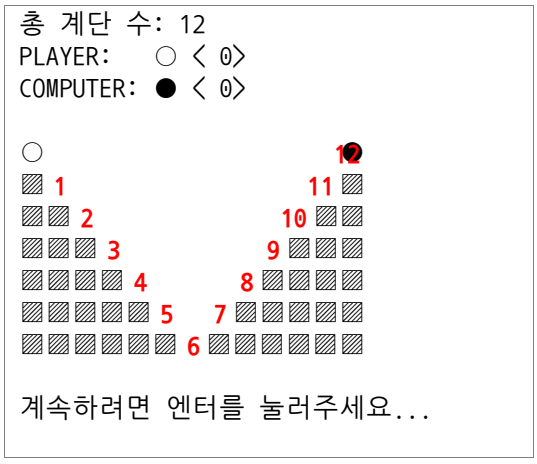
◆ **그림 2의 (2)**

PLAYER는 왼쪽 계단 위에서 COMPUTER는 오른쪽 계단 위에서 게임을 시작한다. PLAYER는 '○', COMPUTER는 '●' 기호로 현재 위치를 나타내고 있다. 현재는 둘 다 이동한 계단수가

0인 상태이다. 게임 도중에 같은 계단에 서게 되면 기호 '◑'로 표시하도록 한다.
게임을 위해 이동해야 할 계단의 수는 홀수일 때와 짝수일 때의 구성이 다르니 그림 3과 Appendix의 그림 17을 참고하여 작성하도록 한다. 그림 3의 빨간색으로 표시된 숫자는 플레이어가 이동하게 될 순서이며 플레이어가 현재 위치에서 한 계단을 이동하게 되면, 1이라고 적힌 위치로 이동하게 된다. 플레이어와는 반대로 컴퓨터는 현재 위치에서 한 계단을 이동하게 되면 그림 3의 (a)의 경우 10, 3의 (b)의 경우 11이라고 적힌 위치로 이동하게 된다.



(a) 총 계단수가 홀수인 경우



(b) 총 계단수가 짝수인 경우

그림 3. 총 계단 수 예시 화면

- 계단을 출력하기 위해서 다음과 같은 함수를 정의한 후 사용한다.
  - **print_stairs()**: 컴퓨터와 플레이어의 현재 이동한 계단 수와 이동해야 할 총 계단 수를 매개변수를 전달받아 화면에 현재 진행상황을 출력한다. 승부가 결정될 때마다 해당함수를 호출하여 현재 진행상황을 지속적으로 출력한다. **(중첩 for문을 사용하여 구현할 것)**

그림 3의 상태에서 엔터를 누르면 화면 지우기 후 그림 4와 같이 가위, 바위, 보 중 하나를 선택하는 화면으로 넘어간다. 플레이어는 '가위', '바위', '보' 중 하나를 선택하여 입력한다. 선택 범위를 벗어나는 입력에 대한 실행 예시는 그림 14를 참조하도록 한다.

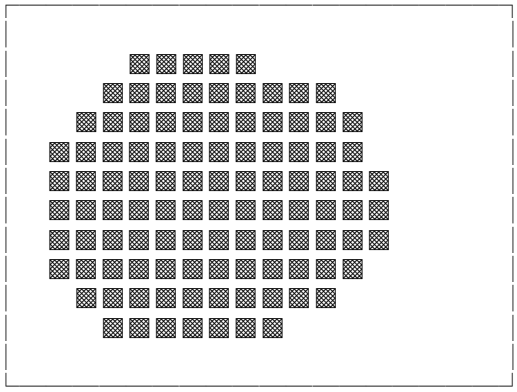
컴퓨터는 random 모듈의 함수를 사용하여 '가위', '바위', '보' 중 하나의 후보를 선택하게 된다. (단, 컴퓨터는 프로그램을 실행할 때마다 랜덤하게 가위바위보를 내도록 구현해야 한다.)

- 컴퓨터의 선택을 생성하기 위해서 다음과 같은 함수를 정의한 후 사용한다.
  - **computer_choice()**: '가위', '바위', '보' 중 무작위로 하나의 후보 선택 후 return

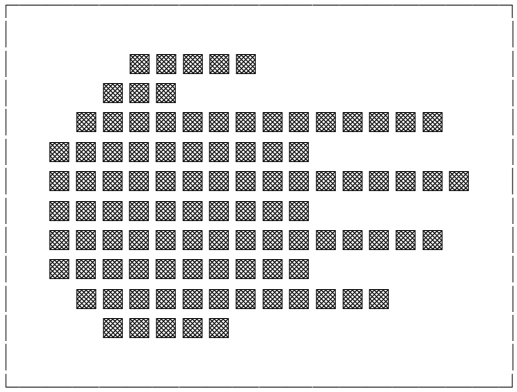
플레이어의 선택 후 그림 4와 같이 플레이어와 컴퓨터가 각각 무엇을 냈는지 출력해 준다.
그림 4는 컴퓨터가 바위, 플레이어가 보를 선택한 후 출력한 예시, 그림 5는 컴퓨터와 플레이어 모두 가위를 선택한 후 출력한 예시이다.

[공격권 결정 가위바위보]
가위, 바위, 보 중 하나 선택: 보

[컴퓨터 선택]



[플레이어 선택]



[결과] 플레이어 공격, 컴퓨터 수비입니다.

계속하려면 엔터를 눌러주세요...

그림 4. 공격권 결정을 위한 가위바위보 - 플레이어 승리 예시

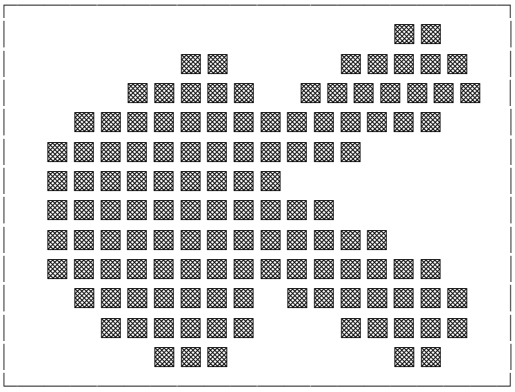
- 가위바위보 출력을 위해 아래의 함수를 정의하고 사용한다.
  - **print_scissors()**
    그림 5의 [컴퓨터 선택] 출력과 같이 가위 이미지를 출력
  - **print_rock()**
    그림 4의 [컴퓨터 선택] 출력과 같이 바위 이미지를 출력
  - **print_paper()**
    그림 4의 [플레이어 선택]의 출력과 같이 보 이미지를 출력

그림 4와 같이 가위바위보에서 승부가 가려진 경우 "[결과] 플레이어 공격, 컴퓨터 수비입니다." 혹은 "[결과] 컴퓨터 공격, 플레이어 수비입니다." 를 출력하고 엔터 입력을 기다린다.

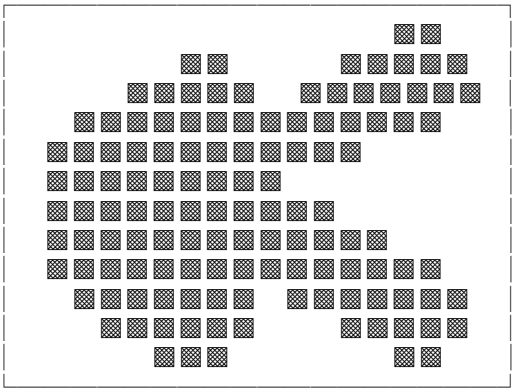
만약 '공격권 결정 가위바위보'에서 승부가 가려지지 못한 경우 그림 5와 같이 "[결과] 무승부입니다." 를 출력한 후 엔터 입력을 기다린다. 엔터 입력이 들어온 경우 화면을 지우고 위의 과정을 승부가 결정될 때까지 반복한다.

[공격권 결정 가위바위보]
가위, 바위, 보 중 하나 선택: 가위

[컴퓨터 선택]



[플레이어 선택]



[결과] 무승부입니다.

계속하려면 엔터를 눌러주세요...

그림 5. 공격권 결정을 위한 가위바위보 - 무승부 예시

그림 4와 같이 공격과 수비가 결정된 상태에서 엔터 입력이 들어온 경우 화면을 지우고 그림 6과 같이 다시 가위바위보 입력을 받는다. 이 때 화면 최상단에 승리 시 이동 칸 수를 출력하도록 한다. 승리 시 이동 칸 수는 처음에 1로 시작하며, 승부가 나지 않을 때마다 1씩 증가한다.

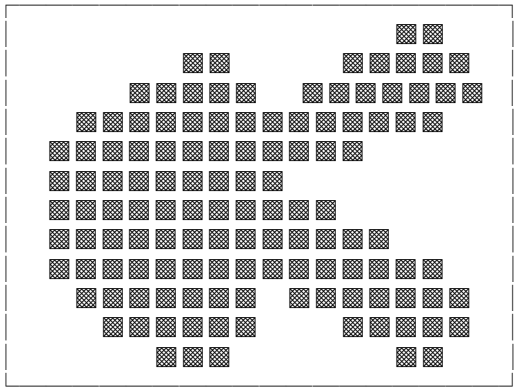
```
[묵찌빠]
승리 시 이동 칸 수: 1
플레이어 공격, 컴퓨터 수비입니다.
가위, 바위, 보 중 하나 선택:
```

그림 6. 공격/수비 결정 후 화면 입력 대기

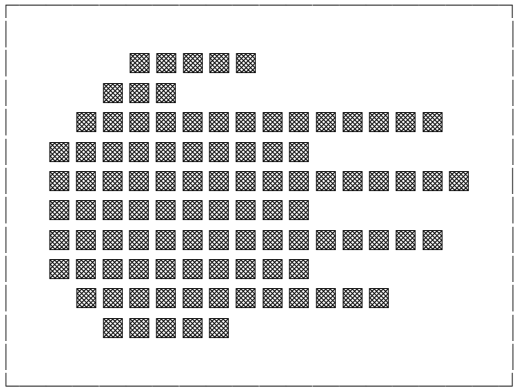
컴퓨터는 앞서 이야기한 **computer_choice()** 함수를 활용하여 가위, 바위, 보 중 무작위로 하나를 선택하도록 한다. 플레이어의 선택 후 승부가 갈리지 않았다면 현 상태에서 묵찌빠 규칙 상 이긴 사람이 공격권을 가져가고 승부가 날 때까지 위의 과정을 반복한다. 그림 7은 플레이어가 공격권을 가진 상태에서 공격에 실패한 후 컴퓨터가 공격권을 가져갔을 때의 출력 예시로, 하단에 "컴퓨터 공격, 플레이어 수비입니다." 가 출력되었다.

[묵찌빠]
승리 시 이동 칸 수: 1
플레이어 공격, 컴퓨터 수비입니다.
가위, 바위, 보 중 하나 선택: 보

[컴퓨터 선택]



[플레이어 선택]



[결과] 컴퓨터 공격, 플레이어 수비입니다.

계속하려면 엔터를 눌러주세요...

그림 7. 플레이어 공격 실패 후 컴퓨터 공격권인 상태인 예시

승부가 결정나지 않으면 승부가 날 때까지 위의 과정을 반복하되, 매 턴마다 '승리 시 이동 칸 수'를 1씩 증가시킨다.

그림 8은 그림 7의 상태에서 엔터를 입력한 후의 예시로 화면을 지운 후 다시 가위바위보 입력을 기다리고 있는 상태이다. '승리 시 이동 칸 수'가 그림 7과 달리 1에서 2로 증가한 것을 확인할 수 있고 컴퓨터가 공격권을 가진 상태임을 확인할 수 있다.

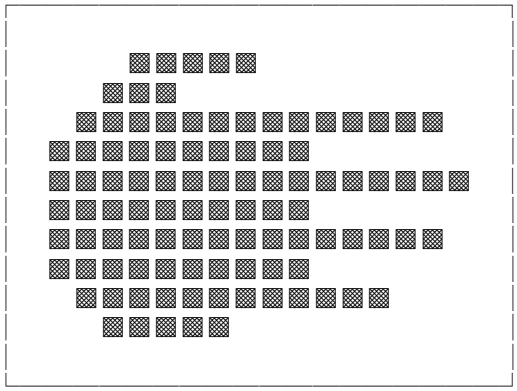
[묵찌빠]
승리 시 이동 칸 수: 2
컴퓨터 공격, 플레이어 수비입니다.
가위, 바위, 보 중 하나 선택:

그림 8. 그림 7의 상태에서 엔터를 입력한 후 화면 입력 대기

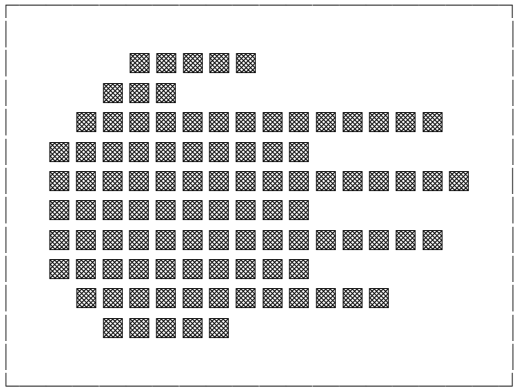
그림 9와 같이 컴퓨터와 플레이어가 같은 선택을 하여 승부가 결정되었을 때 묵찌빠를 종료한다. 그림 9는 플레이어가 공격권을 가진 상태에서 플레이어와 컴퓨터가 모두 '보'를 선택했을 때의 예시이며 5턴만에 승부가 결정되어 플레이어가 5칸을 이동하게 되는 예시이다.

[묵찌빠]
승리 시 이동 칸 수: 5
플레이어 공격, 컴퓨터 수비입니다.
가위, 바위, 보 중 하나 선택: 보

[컴퓨터 선택]



[플레이어 선택]



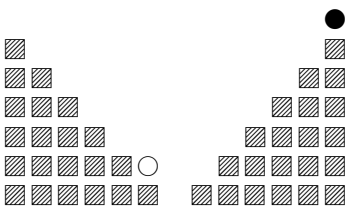
[결과] 묵찌빠 종료
플레이어 승, 5 칸 이동합니다.

계속하려면 엔터를 눌러주세요...

그림 9. 플레이어 승리로 승부가 결정되었을 때의 예시

그림 9와 같이 승부가 결정된 상태에서 플레이어가 엔터를 입력하면 화면 지우기 후 바로 다음 판을 진행한다. 이때, 플레이어와 컴퓨터의 현재 계단에서의 위치를 숫자와 그림으로 그림 10과 같이 표시한다. 5턴만에 승부가 결정되었기 때문에 앞에서 명시한 계단 이동 규칙을 적용하여 플레이어가 5칸 이동하게 되었음을 확인할 수 있다.

총 계단 수: 12
PLAYER: ○ < 5>
COMPUTER: ● < 0>

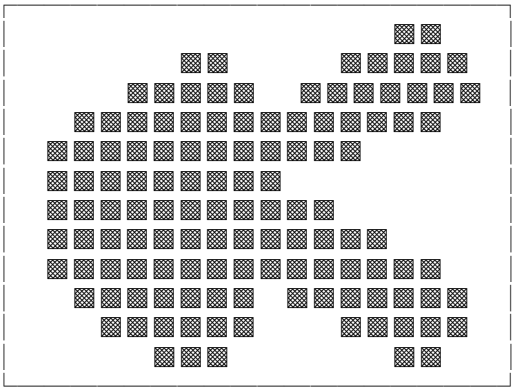


계속하려면 엔터를 눌러주세요...

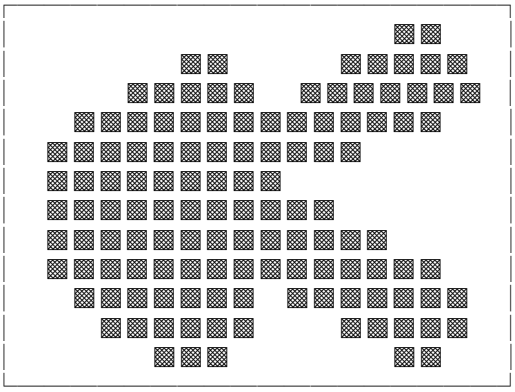
그림 10. 그림 2의 상태에서 그림 9와 같이 플레이어가 승리 후 5칸 이동하였을 때의 예시

그림 11 및 12는 그림 10의 상태에 이어 컴퓨터가 4턴만에 승리한 경우의 예시이다. 플레이어의 위치 이동 5, 컴퓨터의 위치 이동 0인 상태에서 컴퓨터가 4턴만에 이겨 컴퓨터가 4계단 이동하게 되었다. 승부의 결과는 그림 11과 같이 "컴퓨터 승, 4칸 이동합니다"가 출력된다. 플레이어가 엔터를 입력하면 화면을 지우고 그림 12의 예시처럼 승부의 결과가 반영된 결과를 출력한다.

```
[묵찌빠]
승리 시 이동 칸 수: 4
컴퓨터 공격, 플레이어 수비입니다.
가위, 바위, 보 중 하나 선택: 가위

[컴퓨터 선택]
```



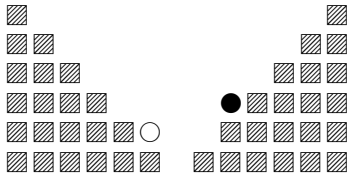
```
[플레이어 선택]
```



```
[결과] 묵찌빠 종료
컴퓨터 승, 4칸 이동합니다.

계속하려면 엔터를 눌러주세요...
```

그림 11. 컴퓨터 승리로 승부가 결정되었을 때의 예시

```
총 계단 수: 12
PLAYER:   ○ < 5>
COMPUTER: ● < 4>
```



```
계속하려면 엔터를 눌러주세요...
```

그림 12. 컴퓨터 승리 후 4칸 이동하였을 때의 예시

그림 13은 게임 중간에 플레이어와 컴퓨터가 같은 계단 위치에 서게 된 경우로 기호 '◑'로 위치가 표시됨을 보여준다.

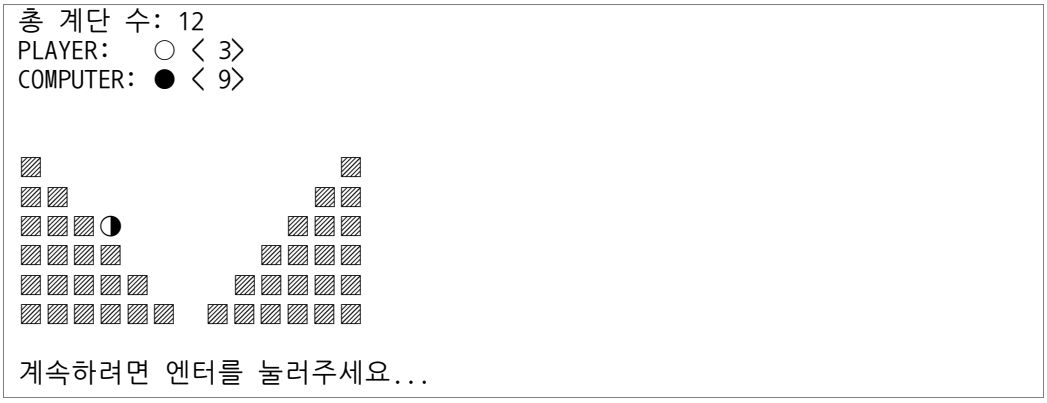


그림 13. 게임 중간에 플레이어와 컴퓨터의 위치가 같은 경우

플레이어가 가위, 바위, 보에서 벗어난 후보를 선택하여 입력하면 특별한 에러 메시지 없이 그림 14처럼 정상적인 입력을 받을 때까지 다시 입력을 받도록 한다.

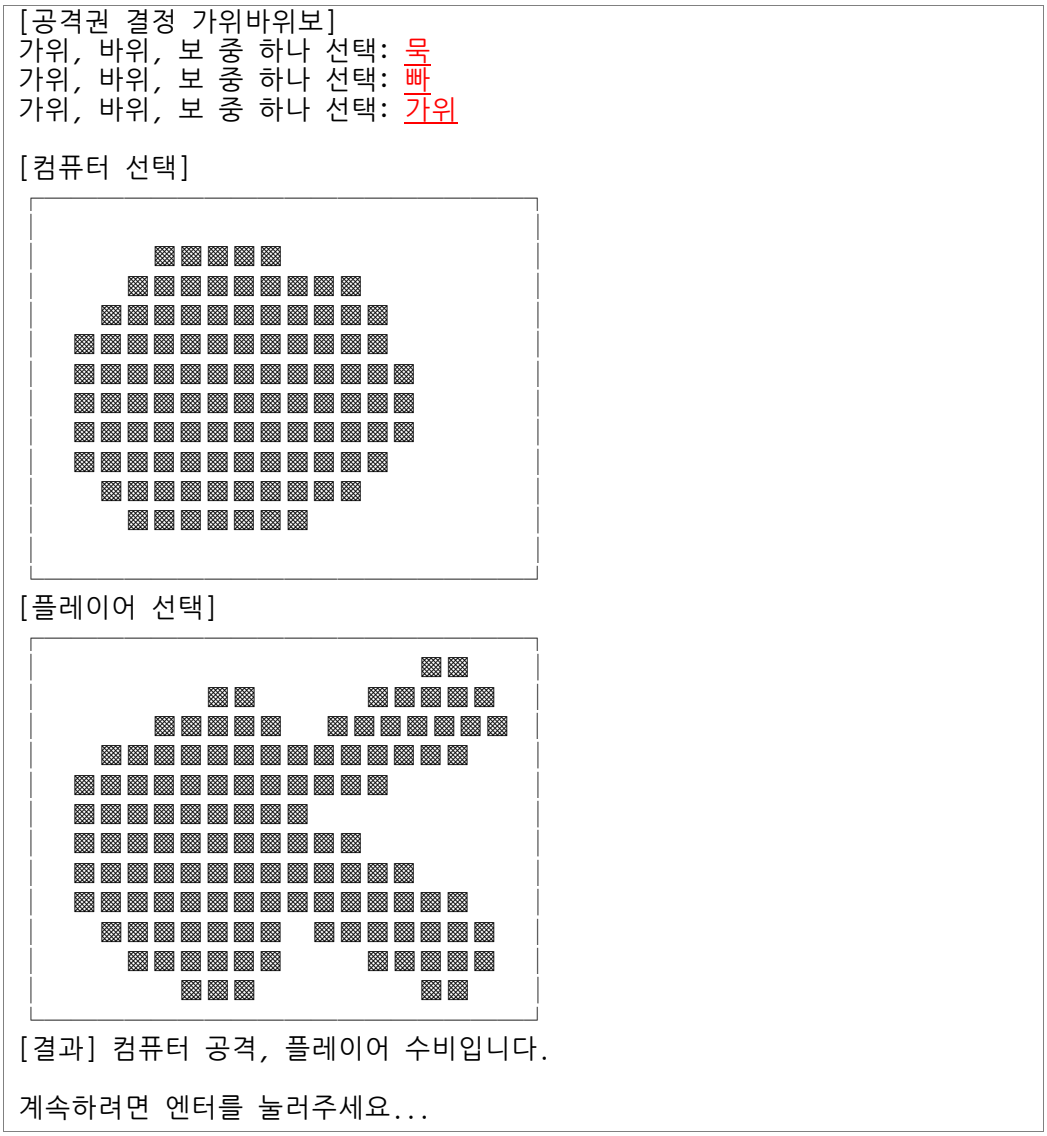


그림 14. 플레이어의 잘못된 입력 예시

II. 게임 종료

게임이 끝나는 경우는 컴퓨터 혹은 플레이어가 상대방의 시작점까지 도달한 경우이다.

그림 15의 왼쪽 예시는 12개의 계단 중 플레이어와 컴퓨터 모두 11번째 계단에 위치한 상태의 예시이다. 그림 15의 왼쪽 상태에서 플레이어가 2턴만에 승리를 가져와 2계단을 이동하게 되었다. 플레이어는 11번째 계단에서 2계단을 이동해야 하지만 총 12개의 계단이 존재하므로 플레이어의 최종 위치는 12번째 계단이 되고 이는 컴퓨터의 게임 시작위치이다. 그리고, 그 아래 그림 15의 오른쪽 예시와 같이 "플레이어 최종 승리!!!"를 출력 후 게임을 종료한다.

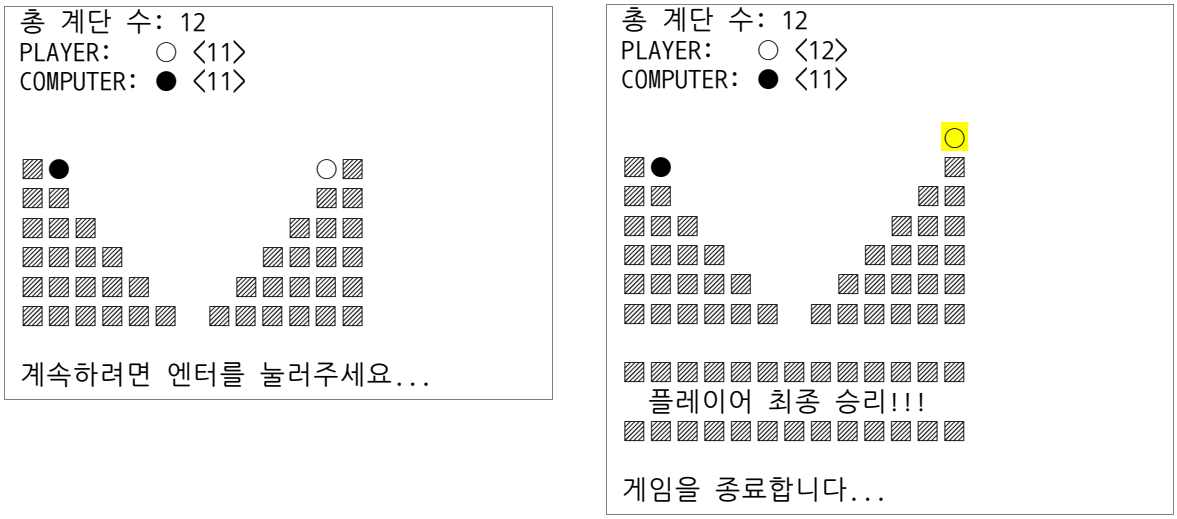


그림 15. 게임 종료 예시 (플레이어 승) (플레이어 최종 승리 위치 수정)

그림 16은 컴퓨터가 먼저 플레이어의 시작 위치에 도착한 경우로 "컴퓨터 최종 승리!!!" 출력 후 프로그램을 종료한다.

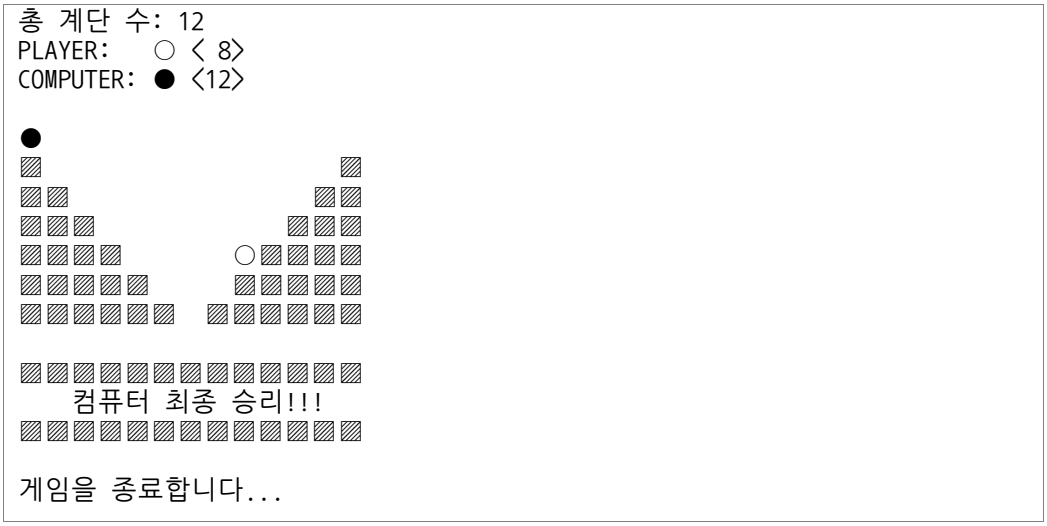


그림 16. 게임 종료 예시 (컴퓨터 승)

Tips

■ 화면 지우기

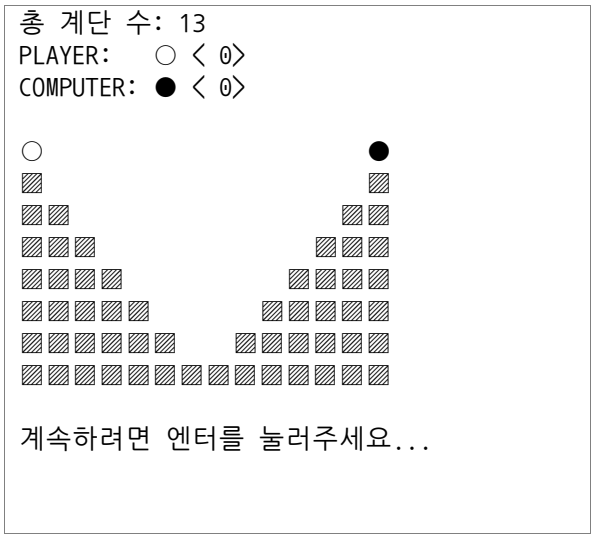
본 과제에서는 화면 지우기를 활용해야 하는 경우가 많다. 그러나 jupyter notebook, windows/linux 등 개발 환경 및 실행 차이로 인해 사용해야 하는 화면 지우기 명령어가 환경에 따라 달라질 수 있다. **본 과제에서는 아래와 같이 화면 지우기를 위해 clear_screen() 함수를 만들어 사용하도록 하며, 이를 따르지 않을 경우 감점된다.**

Windows/Linux 환경인 경우 os module을 사용하여 현재 콘솔 창에 출력된 내용을 지울 수 있다. jupyter notebook 환경인 경우 IPython module을 사용하여 현재 출력된 내용을 지울 수 있다. 아래 코드를 환경에 맞게 만들어 실행해본 뒤 clear_screen() 함수 부분을 주석 처리하여 실행해봄으로써 차이를 파악하도록 한다.

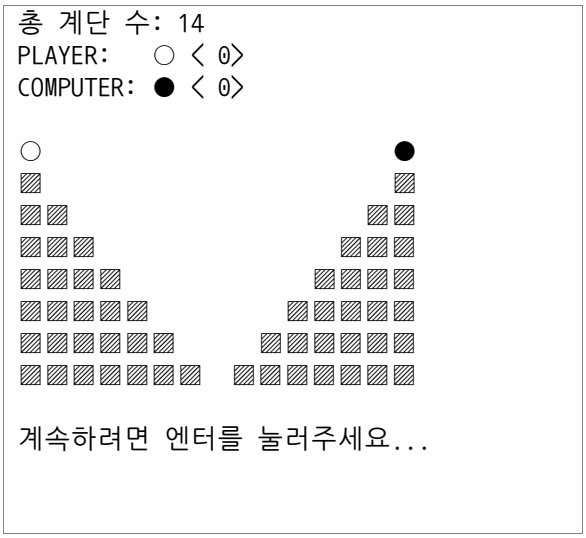
```
import os
from IPython.display import clear_output

def clear_screen():
    # os.system('cls')   # Windows 콘솔 창에서 실행 시 주석 해제
    # os.system('clear') # Linux 콘솔 창에서 실행 시 주석 해제
    clear_output()       # jupyter notebook 외 실행 시 주석 처리할 것
    return

print('Hello, World!')
clear_screen() # 해당 부분 주석처리한 후 실행하여 결과 비교
print('Hello, World!')
```

Appendix

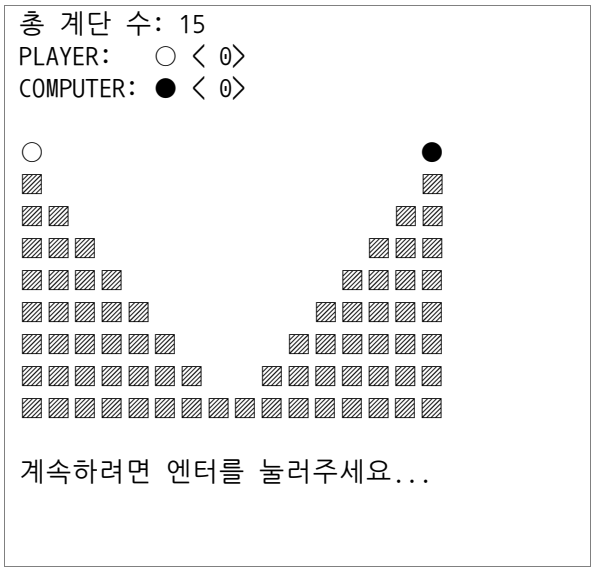
■ 계단의 개수 예시



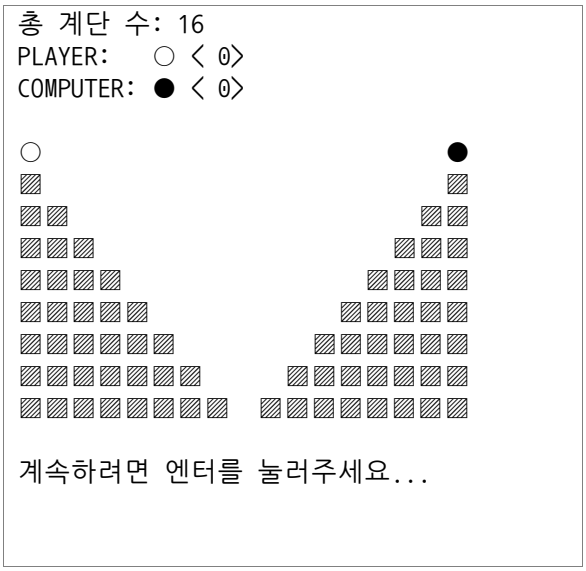
(a) 총 계단수가 13개인 경우



(b) 총 계단수가 14개인 경우



(c) 총 계단수가 13개인 경우



(d) 총 계단수가 14개인 경우

그림 17. 총 계단 수 예시 화면

그림 17은 계단의 개수가 증가하면서 계단 출력이 어떻게 변화하는지에 대한 예시이다. 가운데 칸이 가장 낮은 위치이고 가장자리로 갈수록 층이 높아지는 V자 형태를 띄고 있음을 확인할 수 있다.