2023년 2학기

프로그래밍과 문제해결

Assignment #2

담당교수: 윤은영
학번: 20230673
학과: 무은재학부
이름: 전재영

명예서약(Honor code)

"나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다."

Problem: 요트 다이스 (Yacht dice)

1. 문제의 개요

본 프로그램을 간략히 설명하면 다음과 같다.
- 랜덤으로 주사위 5개를 굴린다.
- 원하는 주사위 이외의 주사위를 2번까지 다시 굴릴 수 있다.
- 12개의 카테고리에 맞는 족보를 선택해 점수를 획득하고, 기록한다.
- 유저와 컴퓨터가 번갈아 가면서 위 과정을 수행하며, 최종적으로 점수가 큰 사람이 승리한다.
- 중간에 진행 상황을 .txt 파일로 저장할 수 있다.
- 저장된 .txt 파일을 이용해 불러오기할 수 있다.

2. 알고리즘

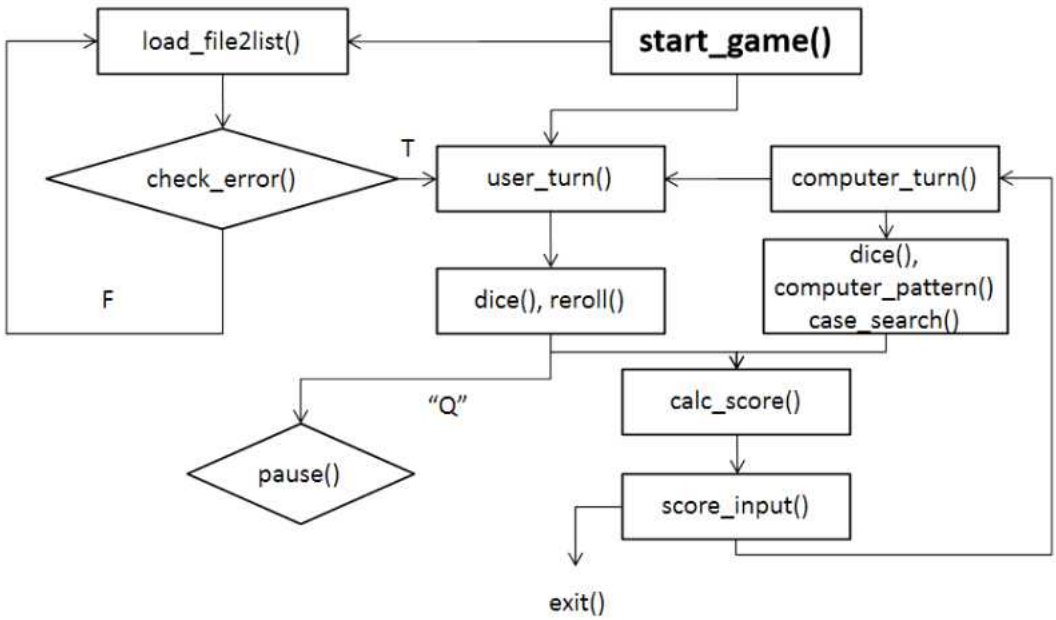
본 프로그램 작성을 위한 알고리즘을 Pseudo 코드 형태로 나타내면 다음과 같다.

Pseudo-algorithm for game
  // 프로그램에 필요한 변수들은 미리 선언해놓은 것으로 가정한다.

```
1 include random, os, sys
2 get user input -> 1 : start_game()
3                   -> 2 : load_game()
4                   -> 3 : exit_game()
5 start_game() : while(game end) : ( user_turn() ; com_turn() )
6                 if game_end : break
7 user_turn() : dice() -> reroll() ; calc_score() ; score_input()
8 com_turn() : com_selection ; calc_score() ; score_input()
9 dice() : return random int (1~6) of 5 dice
10 reroll() : get user's input and reroll dice at input
11 calc_score() : from list of dice -> calc the score and category's index
12 score_input() : write the score, used category
13 com_selection() :
14       calc the score all of the possible category
15       return the largest category and score
16 pause() : from game's log, write the .txt file
17 load_file2list() :
18       check the check_error() -> True # 오류가 없을 때
19       from user's input -> load file -> list
20 check_error() :
21       check for line numbers, number, int, float
```

22 if everything is fine, -> return True ; else -> Fals
23 exit_game() : break
24
25 start_game()

위의 의사 알고리즘을 Flowchart를 통해 표현하면 다음과 같습니다.



3. 프로그램 구조 및 설명

먼저 프로그램을 이루는 주요 변수들과 함수들을 설명하고, 구조를 설명하겠습니다.

변수 :
upper_game_index : "1"~"6"에 해당하는 카테고리가 사용됬는지를 나타내는 변수이다. 이중 리스트로 구현했으며, [0]은 각 리스트의 이름, [1]은 유저의 카테고리, [2]는 컴퓨터의 카테고리 사용현황을 나타낸다.
upper_game_index_score : "1"~"6"의 해당하는 점수를 나타낸다. 이중 리스트로 구현했으며, [0]은 유저의 점수, [1]은 컴퓨터의 점수를 나타낸다.
player/computer_upper_sum : "1" ~ "6"의 점수의 합을 나타낸다.
user/computer_bonus : upper_sum 이 63 이상일 때 35점이 되는 보너스 점수이다.
lower_game_index(_score) : "C" ~ "Y"의 해당하며, upper와 같은 방식이다.
user/com_total : 유저와 컴퓨터의 총 합산 점수를 나타낸다.
user/com_cnt : 유저와 컴퓨터의 진행한 턴수를 나타낸다.

함수 :

start() : 시작해서 명령을 받습니다.
start_game() : 게임의 초기 설정을 하고, 유저와 컴퓨터의 턴을 반복하는 함수입니다.
print_start_screen() : 단순히 시작 화면을 출력합니다.
print_score_board() : 점수판을 출력합니다.
user/computer_turn() : 각각 유저의 턴과 컴퓨터의 턴을 진행하는 상황입니다.
is_bonus_user/computer : 유저나 컴퓨터의 upper_sum 이 63 이상이라면 보너스 점수 35를 리턴하는 함수입니다.
random_dice() ; 랜덤한 주사위를 리스트의 형태로 리턴합니다.
reroll() : 유저의 입력을 받아 최대 2번까지 주사위를 다시 돌리고, 유저가 선택한 카테고리와 최종적인 주사위를 반환합니다.
calc_score() ; 주사위를 굴린 결과와 그 인덱스를 통해 그 점수를 반환합니다.
score_input() : 점수를 변수에 기록하는 함수입니다.
computer_pattern() : 컴퓨터가 주사위를 다시 굴리는 과정입니다.
computer_case_search() : 이미 선택된 카테고리를 제외하고 최댓값의 인덱스를 반환합니다.
pause() : “q”/“Q” 입력을 받으면 게임의 진행 상황을 기록해 txt 파일로 저장합니다.
load_file2list() : 파일을 열어서 변수에 저장하고 리턴합니다.
check_error() : 파일을 열 때 가능한 파일 형식인지 판단합니다.

구조도에 대한 설명은 다음과 같습니다.

a-1) 게임의 시작

- 프로그램을 시작하면 유저의 입력을 통해 명령을 받습니다. 만약 1번의 명령을 입력할 경우, start_game()을 통해 바로 게임을 시작합니다. 만약 2번의 명령을 입력한 경우, load_file2list()를 이용해 초깃값을 불러와서 게임을 시작합니다. 3번의 명령을 입력한 경우, 게임을 종료합니다.

a-2) load_file2list()

- 파일을 불러올 때, check_error()를 통해, 불러올 수 있는 파일인지 확인하는 과정이 필요합니다. 불가능한 경우는, **1.** txt 파일의 줄이 12줄을 넘을 때, (단, 공백만 있는 경우는 제외) **2.** score의 값이 소수일 때, **3.** “1” ~ “6” 의 카테고리 I에서 점수가 1*i ~ 5 * i 의 값이 아닐 때, **4.** “C”, “4K”, “FH”의 점수가 5이상 30이 아닐 때, (FH의 경우 6과 29가 불가능), **5.** “SS, ”LS“, ”Y“의 점수가 각각 15, 30, 50이 아닐 때, 불가능한 경우로 판단하고, 다시 파일명 입력을 받습니다.

b-1) 게임 진행 : 유저의 턴

- 게임을 시작하면, print_score_board()를 통해 점수판을 출력하고, 주사위 5개를 굴린 결과를 출력합니다. 그 후, reroll() 함수에 따라 주사위를 다시 굴릴 수 있습니다. try except 문을 사용해 엔터만 입력 했을 때, 범위를 벗어나는 입력을 받았을 때, 정수가 아닌 값을 받았을 때, q를 입력해 저장하려 했을 때의 경우를 처리했습니다.
- 주사위를 굴린 후, 원하는 카테고리를 선택합니다. 그 후, calc_score()를 통해 선택한 카테고리와 주사위에 해당하는 점수를 리턴받고, score_input()을 통해 변수에 기록합니다.

b-2) 게임 진행 : 컴퓨터의 턴

- 컴퓨터의 턴에서는 유저와 비슷한 방식으로 처음엔 주사위 5개를 굴린 결과를 출력합니다.

그 후, computer_patten을 이용해, 주사위의 패턴에서 가능한 모든 카테고리의 점수를 확인하고, 가장 큰 값의 카테고리를 고른 후, 주사위를 다시 굴리는 과정을 진행합니다. 그 후, 그 가장 큰 값의 카테고리의 점수를 리턴받고 기록합니다.

- 유저와 턴과 컴퓨터의 턴을 턴 진행 수가 12가 넘을 때까지 반복합니다.

c) 게임의 저장

- 유저의 입력 상황에서 "q"또는 "Q"를 입력하면 유저가 입력한 이름으로 게임의 진행 상황을 저장해 txt 파일로 만듭니다.

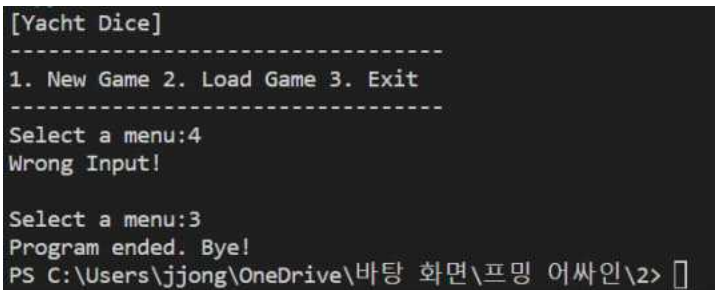
d) 게임의 종료

- 턴수가 12번을 넘으면, 유저와 컴퓨터의 total_score를 비교해 유저가 크면 win, 같으면 draw, 작으면 lose를 출력하고, 다시 start()를 실행해 게임을 다시 진행할지, 아니면 종료할지 선택할 수 있게 합니다.
- 만약 3번을 통해 게임을 종료하려 할 경우, 프로그램을 정상적으로 종료합니다.

4. 프로그램 실행방법 및 예제

- vscode를 이용해 제작과 실행을 했습니다.

[예제 1]



초기 메뉴에서 범위를 벗어난 입력과 종료의 예시입니다.

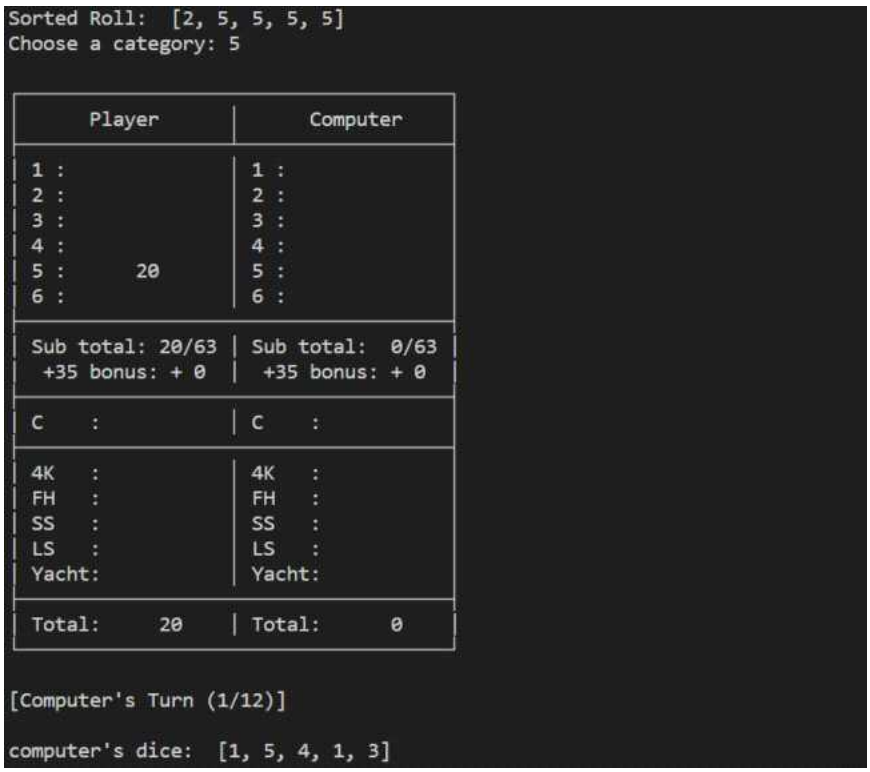
[예제 2]

초기 메뉴에서 1. New Game을 선택한 경우입니다.

[예제 3]

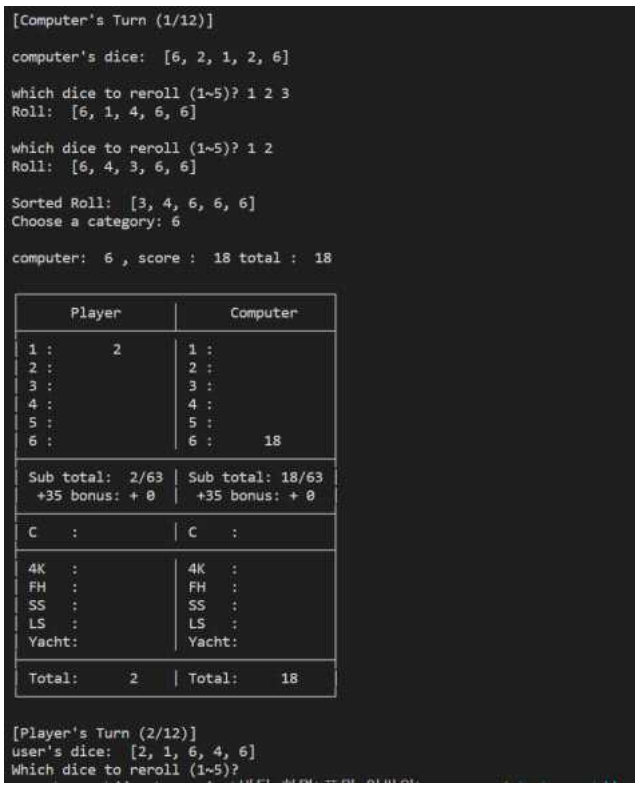
유저가 주사위를 다시 굴리는 상황입니다. 정수 이외의 입력, 범위를 벗어나는 입력을 제대로 처리한 것을 확인할 수 있습니다.

[예제 4]



유저의 점수판 선택 과정입니다.

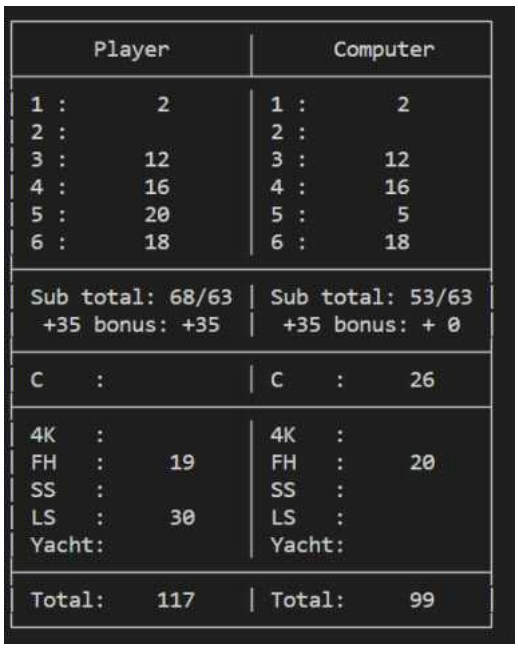
[예제 5]



```
[Computer's Turn (1/12)]

computer's dice:  [6, 2, 1, 2, 6]

which dice to reroll (1~5)? 1 2 3
Roll:  [6, 1, 4, 6, 6]

which dice to reroll (1~5)? 1 2
Roll:  [6, 4, 3, 6, 6]

Sorted Roll:  [3, 4, 6, 6, 6]
Choose a category: 6

computer:  6 , score :  18 total :  18

┌──────────────────┬──────────────────┐
│      Player      │     Computer     │
├──────────────────┼──────────────────┤
│ 1 :       2      │ 1 :              │
│ 2 :              │ 2 :              │
│ 3 :              │ 3 :              │
│ 4 :              │ 4 :              │
│ 5 :              │ 5 :              │
│ 6 :              │ 6 :      18      │
├──────────────────┼──────────────────┤
│ Sub total:  2/63 │ Sub total: 18/63 │
│  +35 bonus: + 0  │  +35 bonus: + 0  │
├──────────────────┼──────────────────┤
│ C    :           │ C    :           │
├──────────────────┼──────────────────┤
│ 4K   :           │ 4K   :           │
│ FH   :           │ FH   :           │
│ SS   :           │ SS   :           │
│ LS   :           │ LS   :           │
│ Yacht:           │ Yacht:           │
├──────────────────┼──────────────────┤
│ Total:      2    │ Total:     18    │
└──────────────────┴──────────────────┘

[Player's Turn (2/12)]
user's dice:  [2, 1, 6, 4, 6]
Which dice to reroll (1~5)?
```

컴퓨터의 턴의 진행을 나타냅니다.

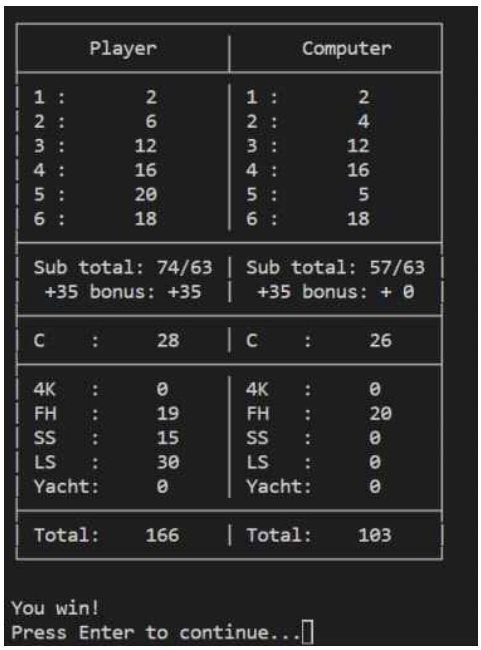
[예제 6]



```
┌──────────────────┬──────────────────┐
│      Player      │     Computer     │
├──────────────────┼──────────────────┤
│ 1 :       2      │ 1 :       2      │
│ 2 :              │ 2 :              │
│ 3 :      12      │ 3 :      12      │
│ 4 :      16      │ 4 :      16      │
│ 5 :      20      │ 5 :       5      │
│ 6 :      18      │ 6 :      18      │
├──────────────────┼──────────────────┤
│ Sub total: 68/63 │ Sub total: 53/63 │
│  +35 bonus: +35  │  +35 bonus: + 0  │
├──────────────────┼──────────────────┤
│ C    :           │ C    :     26    │
├──────────────────┼──────────────────┤
│ 4K   :           │ 4K   :           │
│ FH   :     19    │ FH   :     20    │
│ SS   :           │ SS   :           │
│ LS   :     30    │ LS   :           │
│ Yacht:           │ Yacht:           │
├──────────────────┼──────────────────┤
│ Total:    117    │ Total:     99    │
└──────────────────┴──────────────────┘
```

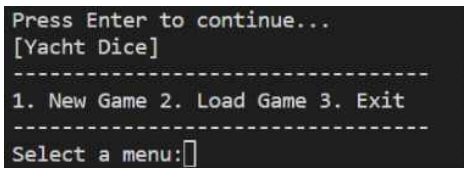
다음과 같이 윗 부분의 점수 합이 63을 넘을 때, 보너스 점수 35점을 얻습니다.

[예제 7]



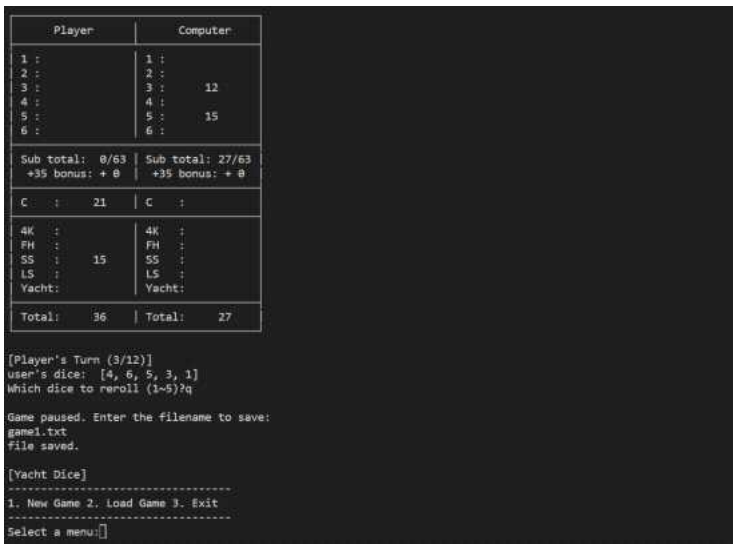
게임의 종료 상황을 나타냅니다.

[예제 8]



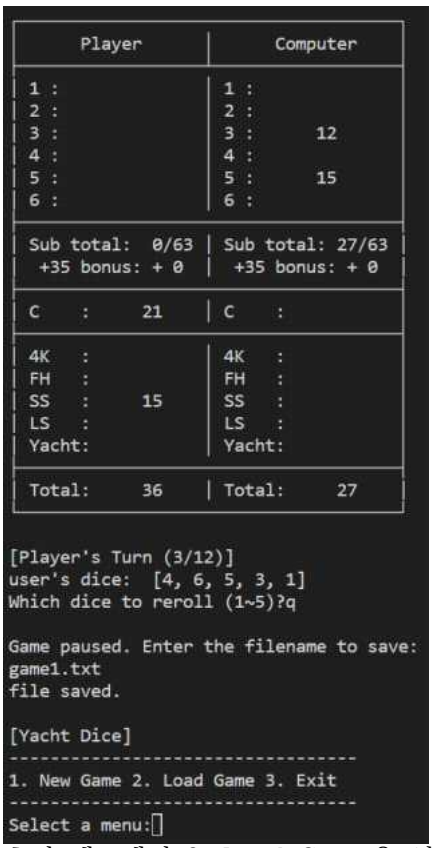
게임이 종료된 후, 엔터를 통해 다시 진행합니다.

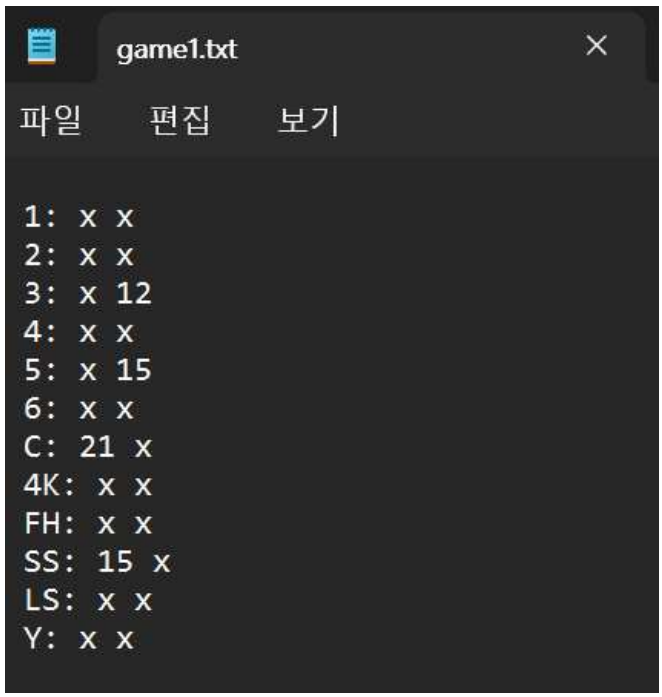
[예제 9]



중간의 q의 입력을 통해 저장할 수 있습니다.

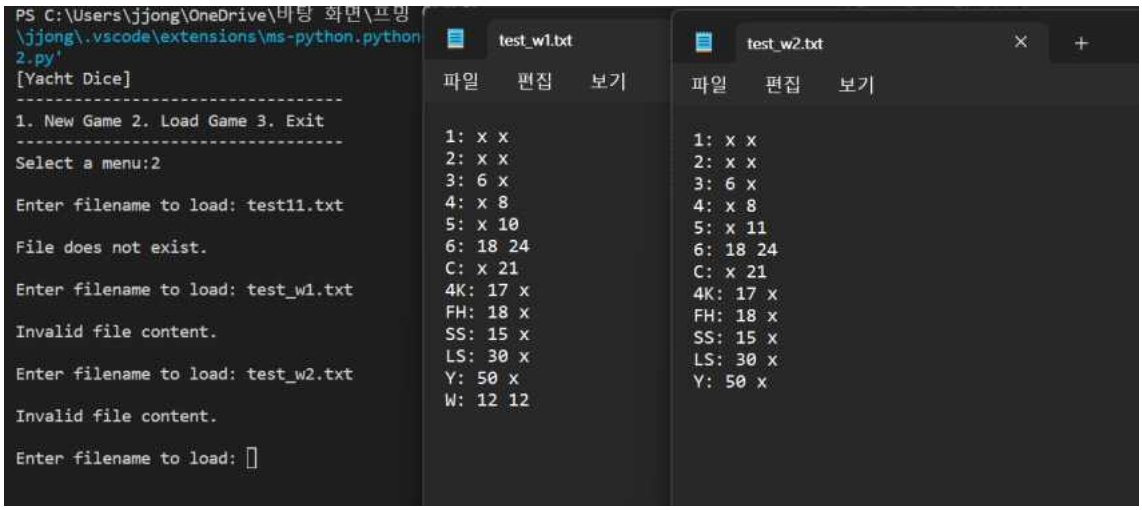
[예제 10]





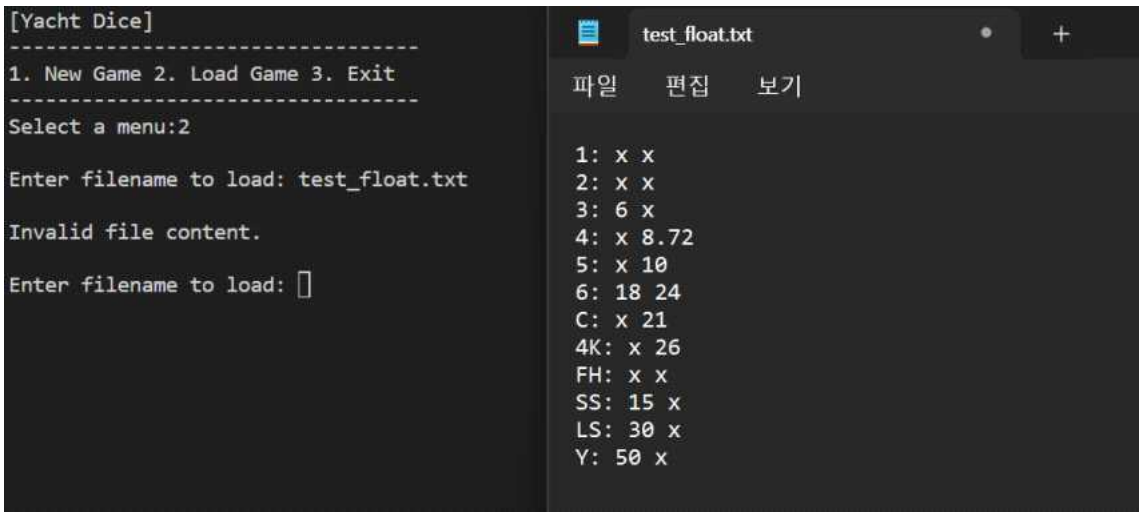
초기 메뉴에서 2. Load Game을 선택한 경우입니다.

[예외 처리-1]



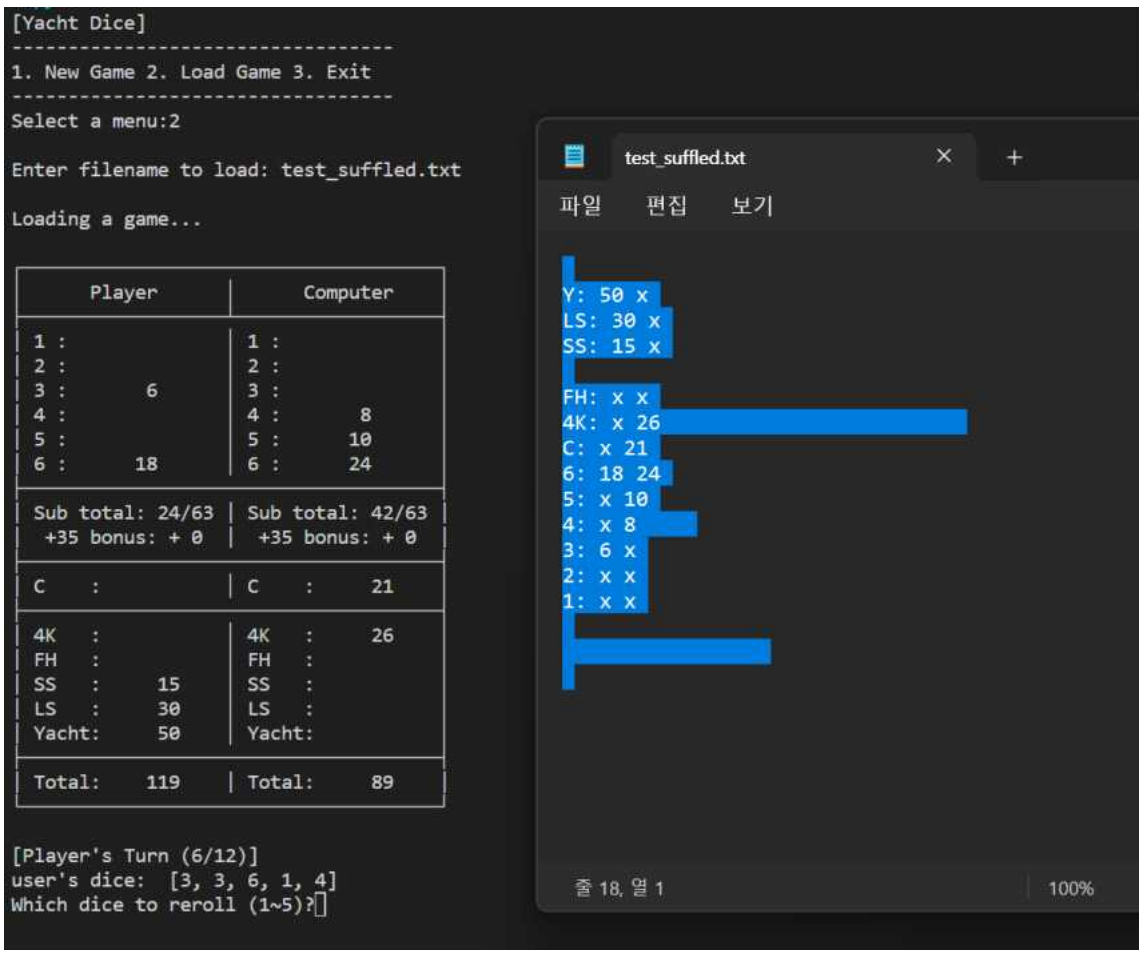
파일이 없는 경우, 총 줄의 개수가 12가 아닌 경우, 주사위 조합으로 불가능한 점수가 적혀 있는 경우의 예외 처리를 하는 모습입니다.

[예외 처리-2]



다음과 같이 float이 있는 경우 역시 예외로 처리합니다.

[예외 처리-3]



다음과 같이 공백, 탭(\t), 줄바꿈(\n)만 있는 경우 정상적으로 실행할 수 있습니다.

5. 토론

- 사용한 모듈은 컴퓨터의 의사결정을 위한 random을 사용했다.
- 점수를 기록하는 과정에서 카테고리를 사용했는지를 나타내기 위해 index를 나타내는 이중 리스트를 만들었다. 만약 사용하지 않은 상태라면, 초기값 0의 상태고,print_score_board()의 출력 과정에서 무시하고 출력된다. 만약 그 카테고리를 사용했다면, index 리스트의 값을 1로 바꾸고, 다시 선택할 수 없게 코드를 구현하였다.
- check_error()에서 float을 예외 처리하는 과정에서 is_integer()를 사용해 만약 0.0 과 같이 float의 자료형이지만 정수에 해당하는 숫자들은 구동이 가능하게 코드를 구현하였다.
- 리스트 자료형의 경우, 함수에서 원본의 값이 변경된 경우, 함수 외부에서도 변경된 값이 유지된다. 그래서 굳이 리턴을 할 필요가 없지만, int와 같은 자료형은 리턴이 필요하므로, 이를 유의해서 코드를 구현하였다.

6. 결론

- 본 과제를 통해 파이썬에서의 함수의 사용과, 이중 리스트 자료형, 파일 입출력에 관한 개념을 익힐 수 있었다. 또한, 12가지의 케이스에 대한 구현을 하고, 일정한 형식에 어긋나는 경우들에 대한 예외 처리에 대해 제대로 배울 수 있었던 것 같다.

7. 개선방향

- 가장 구현에서 어려움을 겪었던 부분은 점수를 이중 리스트로 만들거나, 카테고리를 리스트에 저장할 때, list - out of range 오류를 많이 겪었던 점이다. 이중 리스트를 upper와 lower로 나눠서 구현을 했는데, 이 부분에서 굉장히 비효율적으로 코드가 구현된 것 같다. 그리고 6~11 인덱스를 lower index에 맞게 하기 위해 6을 빼줘야 하는 과정이 필요하는 등 굉장히 아쉬웠다.
- is_bonus_user()와 is_bonus_computer()는 완전히 같은 기능을 수행하고 있어서 굳이 2개의 함수를 만들 필요가 없었는데, 그대로 구현하였다.
- 컴퓨터의 주사위 선택 과정과 유저의 reroll 과정을 따로 구현하였는데, 이것도 조금만 수정을 한다면 컴퓨터도 유저의 reroll을 통해 주사위를 선택할 수 있지 않았을까 생각이 든다. 2번 다시 돌릴 때마다 reroll()을 호출하는 것이 아니라 2번 돌리는 과정을 reroll() 내부에서 수행하기 때문에 수정이 필요할 것 같다.