소프트웨어입문설계 과제#3

~ 0등만 중요한 것은 아니니까요 ~

도입부

지금 와서 하는 말이지만, 과제#2의 목표는 사실 프로그래밍하기 매우 어려운 축에 속합니다. 미래에 누가 상대방으로 오든 무조건 내가 이기게 만들려면 사실상 모든 경우의 수를 다 따져 보아야 하기 때문입니다. 그래서 최초 버전의 hw2_tester.py는 (강사가 일부러 더 느리게 만들어 둔 것도 있긴 하지만) 테스트에 통과하기 위해 매우 많은 대결을 수행해야만 했지요. 그 당시의 강사가 **의도**하지는 않았지만, 이제는 음수 능력치의 가능성도 공개되어 있기에 경우의 수가 사실상 무한에 가깝다 여겨도 될 정도가 되어버렸어요.

목표 자체만 놓고 보았을 때 과제#3은 분명 과제#2에 비해 더 복잡합니다. 이전에는 무조건 내가 0등이기만 하면 되었지만, 과제#3은 캐릭터들이 각각 몇 등에 해당하는지를 모두 판단하여 일렬로 줄을 세워 주어야 하기 때문입니다. 그래도 이번에는 112명의 캐릭터가 모두 공개되어 있으니 불확실한 미래에 대한 걱정은 안 해도 될 것 같군요.

과제 수행 방법

1. 먼저 HY-in 과제 페이지에서 hw3.zip을 다운로드해서 압축을 풀어 주세요.

   (아래 내용들은 각 .py 파일에도 적절한 위치에 TODO 주석으로 달아 두었어요.
   각 단계를 완료했다면 TODO 주석은 지워도 좋아요)

2. 여기에는 .py 파일이 두 개 들어 있습니다. 그 중 hw3_mymodule.py 이름의 mymodule 부분을 자신의 개성이 드러나도록 자유롭게 변경해 주세요. (예: hw3_Racin.py)

3. hw3_core.py를 열고 파일 상단에 있는 import문과 그 아래에 보이는 modules list의 내용을 자신이 방금 바꾼 파일 이름에 맞게 고쳐 주세요.

4. 약간 스크롤 내리면 나오는 `stats`에 주목해 주세요. 어떤 캐릭터들이 있는지, 혹시 내 캐릭터의 이름이 바뀌지는 않았는지 한 번 확인해 보세요.

5. 내 **모듈** 파일로 돌아와서 `duel()`을 구경해 보세요. 과제#2에서와 유사하게 이 **함수**는 두 캐릭터에 대한 정보를 **인수**로 받아 대결 결과를 return해야 하는 **함수**입니다. 지금은 그냥 캐릭터 **이름**만 가지고 승부를 내도록 return문이 적혀 있는 상태군요.

6. 구경이 끝나면 core **모듈**로 가서 F5를 눌러 '제출 전 테스트'를 수행해 봅시다. 적절한 메뉴 메시지를 잔뜩 달아 두었으니 그리 어렵지 않을 거예요. → 실패하는 게 정상임!

7. 이번 버전 테스터는 그리 호락호락하지 않은 것 같습니다. 실패 안내에 강사의 예측 메시지가 같이 나오는 것을 보니 아마 강사는 우리가 이번 과제 하면서 꽤 많이 털릴 것이라 예상하고 있는 듯 합니다. 그러니, 이 설명서 후반부에 있는 '대결 목표 및 규칙'을 읽어 본 다음, 아래 내용을 미리 종이에 적거나 그려 보세요:

   - 내 `duel()`이 어떤 부류의 캐릭터를 어떤 순위권에 둘 것인지
     (예: 정확히 나와 동일한 능력치를 가진 캐릭터를 최상위권에)

   - 각 부류별 특징을 어떻게 감지할 것이며,
     그 부류 안에 속한 캐릭터들 사이의 순위는 어떻게 정할 것인지

   - `duel()`의 **순차** 흐름을 구성할 때 각 부류를 다룰 순서
     (동시에 두 부류에 속하는 캐릭터가 존재한다면 이 순서에 영향을 받게 돼요)

8. 위 내용을 미리 적어 두면 내 `duel()`의 전체적인 **실행** 흐름, 각 부분별 세부 목표를 좀 더 쉽게 파악할 수 있을 거예요. 내 모듈 파일 맨 위에 있는 핵심 주의사항을 수시로 확인하며 본격적으로 과제를 시작해 봅시다!

대결 목표 및 규칙

1. `duel(left, right)`의 **인수**와 return값 설명
   - 두 **인수** `left`, `right`에는 이번에 대결할 두 캐릭터 정보가 각각 담깁니다.
     (과제#2와 달리 이번에는 둘 다 내 캐릭터가 아닐 때가 훨씬 더 많아요)

   - 대결 결과 `left`가 더 우월한 경우 `0`보다 작은 `int` **형식 값**(예: `-1`)을 return해야 합니다.
     대결 결과 `right`가 더 우월한 경우 `0`보다 큰 `int` **형식 값**(예: `1`)을 return해야 합니다.
     무승부는 없습니다.

2. (중요) `duel(left, right)`의 우월성 판정 규칙

   112명 중 임의의 세 캐릭터 `a`, `b`, `c`를 골랐을 때 아래 세 조건을 꼭 만족하도록 구성해야 해요:

   - **수식** `duel(a, b) * duel(b, a) < 0`을 **계산**하면 항상 `True`가 나와야 해요.
     (`a`가 `b`를 이긴다면, `a`가 `left`면 `left` 승, `right`면 `right` 승이 나와야 함)

   - `duel(a, b) < 0 and duel(b, c) < 0`이 `True`라면 `duel(a, c) < 0`도 `True`여야 해요.
     (`a`가 `b`를 이기고 `b`가 `c`를 이기면 `a`는 반드시 `c`를 이겨야 해요. 가위바위보 되면 안 됨!)

   - `a`와 `b`의 능력치가 같다면 `duel(a, c) == duel(b, c)`은 항상 `True`가 나와야 해요.
     (능력치가 같을 때만 이름 비교를 하도록 구성하면 이 조건은 크게 걱정 안 해도 돼요)

   두 번째 조건이 쉽지 않을 수 있어요. 코드 작성 전에 종이로 적어 가며 확인해 봐야 함!

제출 전에 해야 할 일

- (필수)core **모듈**을 F5 눌러서 실행하면 '제출 전 테스트'를 해 볼 수 있어요. 이 테스트를 모두 통과하여 '제출 가능'이 뜬 경우 과제 제출이 가능해요.
  - 정말 희소한 확률로 어쩔 땐 제출 가능 뜨고 어쩔 땐 안 뜰 수 있어요. 자신이 뭔가 다양한 시도를 해 봤다면 넉넉히 네 번 정도 연속 통과하는지 확인해 보면 좋겠어요.
- (선택)제출 전 테스트가 끝나면 '모듈 모아 테스트'를 해 볼 수 있어요. 자신의 **모듈**이 각 캐릭터들을 몇 등으로 간주하고 있는지, 전체 줄세우기 과정이 몇 초 걸리는지 확인해 볼 수 있어요.
- (선택)다른 친구들이 완성한 **모듈**들을 받아서 같은 폴더에 둔 다음, core **모듈**의 import문 부분을 적절히 복붙 / 수정하면 여러 **모듈**들을 다 돌려서 각 캐릭터가 '평균' 몇 등을 하고 있는지 확인해 볼 수 있어요. 친구 **모듈**이 내 캐릭터를 최하위권으로 재고 있다면 제출 전에 진솔한 대화의 장을 마련해 보는 것도 방법일 수 있겠어요.

과제 제출 방법

1. 일단 '제출 전 테스트'를 통과하세요.
2. 아래의 두 파일을 모아 .zip 형식으로 압축하여 HY-in 과제 페이지에 제출해 주세요:
   - hw3_core.py(혹시 모르니 동봉해 주세요. 여러분 코드에 오류가 있을 때 참고하려 함)
   - 여러분이 만든 `duel()`이 담긴 **모듈**(파일)
3. 제출할 때 느낀점도 함께 적어 주세요!
   - 보통 실습 마무리 단계에 썼던 그 칸에다 적으면 돼요.
4. 과제 제출 기한은 4월 24일 화요일 19:29:59(소입설 시험 1초 전)까지입니다!

과제 평가 규칙

이번 과제의 점수는 아래의 조건을 모두 만족하면 100점, 그렇지 않으면 0점입니다:

- HY-in 과제 페이지에 기한 내 제출 성공
- 여러분의 **모듈** 안에 import문, print문이 하나도 없음
- 강사의 Python, 운영체제, 컴퓨터를 공격하려는 '명백한' 시도가 발견되지 않음
  - 이 판단은 강사의 소신을 믿어 주세용. 문제 있는 것 같다 싶으면 개인 연락 하겠음.
- 나누어 준 hw3_core.py 원본으로 **모듈**을 테스트했을 때 '제출 가능!'이 출력됨을 확인

전반부에 여유가 없어서 준비를 결국 못 했는데, 8주차 쉬는 기간동안 시상 요소들 점검하고 후반부 시작할 때 함께 나누어 줄께요.