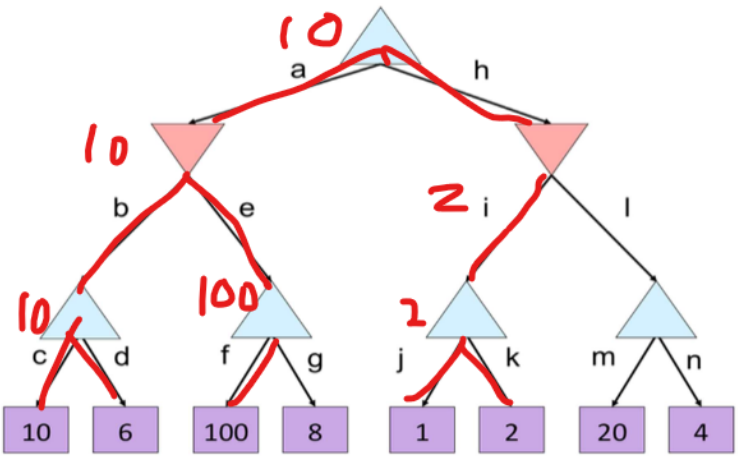
Assignment 2

- capture the result of `autograder.py` in terminal.

```
Finished at 13:45:44

Provisional grades
==================
Question q1: 4/4
Question q2: 5/5
Question q3: 5/5
Question q4: 0/5
Question q5: 0/6
------------------
Total: 14/25
```

1. How can alpha-beta pruning make it more efficient to explore the minimax tree?
    - 알파-베타 프루닝은 가망이 없는(선택될 가능성이 없는) 노드를 탐색하지 않음으로써 더 효율적으로 트리를 탐색할 수 있다. 이를테면, 수업의 예제였던 아래의 사진에서, `g`는 그 값과 무관하게 선택되지 않는데, 값이 `100`보다 크다면 Min-Agent에 의해서 선택되지 않을 것이고, 값이 `100`보다 작다면 Max-Agent에 의해서 선택되지 않을 것이기 때문이다. 마찬가지로, `l`은 그 값과 무관하게 선택되지 않는데, 값이 `2`보다 크다면 Min-Agent에 의해서 선택되지 않을 것이고, 값이 `2`보다 작다면 최상위의 Max-Agent에 의해서 선택되지 않을 것이기 때문이다. 결과적으로 알파-베타 프루닝은 해당 노드들을 탐색하지 않게 되고, 더 효율적으로 트리를 탐색할 수 있다.



2. Is there a situation where the Reflex agent performs better than the minimax or alpha-beta pruning algorithm?
    - 선택이 빠른 시간 내에 이루어져야 하는 경우, Reflex Agent가 Minimax Agent보다 더 적합하다. Reflex Agent는 미래의 상황을 계산하지 않기 때문에 결정에 필요한 오버헤드가 더 적을 것이기 때문이다.
    - 또한, 상대 Agent가 충분히 합리적이지 않다면 Reflex Agent가 더 잘 작동할 수 있다. Minimax Agent는 상대 Agent 역시 Minimax Agent처럼 결과를 최대로 만들 수 있는 선택을 하리라고 가정한다. 하지만 그렇지 않은 Agent를 대상으로는, Minimax Agent가 Reflex Agent보다 잘 작동한다고 보장할 수 없다.
3. Evaluation Function이 많은 함수로 구성될수록 항상 더욱 효과적일까?
    - 반드시 그렇지는 않을 것이다. 더 많은 함수로 구성될수록 결과 Evaluation Function의 값이 교란될 확률이 높아진다. 이를테면, 유령과의 거리를 점수로 취하는 함수와 점과의 거리의 역수를 점수로 취하는 함수가 서로 상충하여 팩맨을 제자리에 머물게 할 수 있다. 이 경우 움직이는 것이 더욱 효과적임에도 팩맨은 움직이지 않을 것이다.