사슴사냥대회

| Background

✓ 기본적인 자료구조와 알고리즘을 이해하고 활용

| Goal

✓ 문제를 정확히 이해하고 자료구조의 활용 및 알고리즘을 설계할 수 있다.

| 파이썬 트랙 제출 방법 및 팀편성 관련

1) 파일 이름 및 제출 방법

- **첨부로 제공되는 main.py 소스 코드를 확인하고 main.py 소스 코드 중 아래 글상자에 표시된 소스 코드 부분을 작성하여 사슴사냥대회_지역_반_조(이름1_이름2)_python.txt 파일을 만들어 제출한다. ※ 파일명 명명 규칙 엄수**

```
# def Me(opp, turn, opp_prev, opp_last_pattern):
#     ToDo
```

(학사시스템 > QUEST 에 업로드)

관통프로젝트 2인 1팀으로 구성하여 협동하여 문제를 풀 것

- **팀 편성 현황은 MM공지 확인 필수!**

3) 당일 16시까지 제출

4) 평가 중 문의사항은 1:1 문의게시판 이용

5) 주의 사항

- **돌아가지 않는 코드는 무조건 Fail 처리**
- **실행 시간이 20초 초과되는 코드 Fail 처리**

참고로 여러분이 제출한 코드들을 모아서 사슴사냥대회를 진행하고 결과를 확인할 예정입니다.

성실과 신뢰로 테스트에 볼 것 (부정 행위시 강력 조치 및 근거가 남음)

※ 소스코드 유사도 판단 프로그램 기준 부정 행위로 판단될 시, 0점 처리 및 학사 기준에 의거 조치 실시 예정

사슴사냥대회

| Background

✓ 기본적인 자료구조와 알고리즘을 이해하고 활용

| Goal

✓ 문제를 정확히 이해하고 자료구조의 활용 및 알고리즘을 설계할 수 있다.

| 자바 및 모바일 트랙 제출 방법 및 팀편성 관련

1) 파일 이름 및 제출 방법

- **첨부로 제공되는 Main.java 소스 코드를 확인하고 Main.java 소스 코드 중 아래 글상자에 표시된 소스 코드 부분을 작성하여 사슴사냥대회_지역_반_조(이름1_이름2)_java.txt**
- **파일을 만들어 제출한다. ※ 파일명 명명 규칙 엄수**

```
//    static Opponent Me=new Opponent(){
//        public int hunt(int opp, int turn, int opp_prev, int opp_last_pattern[][]) {
//            //// ToDo
//        }
//    };
```

(학사시스템 > QUEST 에 업로드)

관통프로젝트 2인 1팀으로 구성하여 협동하여 문제를 풀 것

- **팀 편성 현황은 MM공지 확인 필수!**

3) 당일 16시까지 제출

4) 평가 중 문의사항은 1:1 문의게시판 이용

5) 주의 사항

- **돌아가지 않는 코드는 무조건 Fail 처리**
- **실행 시간이 20초 초과되는 코드 Fail 처리**

참고로 여러분이 제출한 코드들을 모아서 사슴사냥대회를 진행하고 결과를 확인할 예정입니다.

성실과 신뢰로 테스트에 볼 것 (부정 행위시 강력 조치 및 근거가 남음)

※ 소스코드 유사도 판단 프로그램 기준 부정 행위로 판단될 시, 0점 처리 및 학사 기준에 의거 조치 실시 예정

사슴사냥대회

| Background

✓ 기본적인 자료구조와 알고리즘을 이해하고 활용

| Goal

✓ 문제를 정확히 이해하고 자료구조의 활용 및 알고리즘을 설계할 수 있다.

| 임베디드 트랙 제출 방법 및 팀편성 관련

1) 파일 이름 및 제출 방법

- **첨부로 제공되는 main.cpp 소스 코드를 확인하고 main.cpp 소스 코드 중 아래 글상자에 표시된 소스 코드 부분을 작성하여 사슴사냥대회_지역_반_조(이름1_이름2)_C.txt 파일을 만들어 제출한다. ※ 파일명 명명 규칙 엄수**

```
//int Me(int opp, int turn, int opp_prev, int opp_last_pattern[][10])
//{
//            //// ToDo
//}
```

(학사시스템 > QUEST 에 업로드)

2) 관통프로젝트 2인 1팀으로 구성하여 협동하여 문제를 풀 것

- **팀 편성 현황은 MM공지 확인 필수!**

3) 당일 16시까지 제출

4) 평가 중 문의사항은 1:1 문의게시판 이용

5) 주의 사항

- **돌아가지 않는 코드는 무조건 Fail 처리**
- **실행 시간이 20초 초과되는 코드 Fail 처리**

참고로 여러분이 제출한 코드들을 모아서 사슴사냥대회를 진행하고 결과를 확인할 예정입니다.

성실과 신뢰로 테스트에 볼 것 (부정 행위시 강력 조치 및 근거가 남음)

※ 소스코드 유사도 판단 프로그램 기준 부정 행위로 판단될 시, 0점 처리 및 학사 기준에 의거 조치 실시 예정

사슴사냥대회

| 문제

'사슴사냥게임'은 죄수의 딜레마와 더불어 게임이론의 유명한 사례로,

2명이 참가하는 비 제로섬 게임의 일종이다.

다음은 사슴사냥게임의 변형이다.

다음 설명을 잘 읽고 설명에서 요구하는 함수(메소드)를 구현하여 제출하시오.

어느 산속에 사냥꾼 둘이 살고 있었다.

사냥꾼 A와 B는 사전 협의 없이 그날 어떤 짐승을 사냥할지 결정한다.

사슴을 사냥하는 데는 둘의 협력이 필요하고, 토끼를 사냥하는 데는 혼자의 힘으로도 충분하다. 뱀을 사냥하는 데에도 혼자의 힘으로 충분하며, 만약 상대방이 토끼를 사냥했다면 뱀을 이용하여 상대방이 획득한 토끼를 상대방 몰래 훔칠 수 있다.

결과적으로 사슴을 사냥하기로 결정했다면, 상대방도 사슴을 택했을 경우, 사냥에 성공하여 둘 다 50의 이득을 취한다. 하지만, 상대방이 토끼나 뱀을 택했을 경우 나는 아무런 이득도 취할 수 없다. (이때, 상대방이 토끼를 택했다면 20의 이득을 뱀을 택했다면 10의 이들을 취득)

토끼를 사냥하기로 결정한 사냥꾼은 상대방이 사슴이나 토끼를 선택할 경우 20의 이득을 취하지만, 뱀을 취할 경우 아무런 이득도 취할 수 없다.

뱀을 사냥하기로 결정한 사냥꾼은 상대방의 결정에 상관없이 10의 이득을 취할 수 있으며, 상대방이 토끼를 택할 경우 20의 이득을 추가로 획득하여 총 30의 이득을 취할 수 있다.

사슴사냥대회

이 상황을 나타내면 다음 표와 같다.

A \ B	사슴	토끼	뱀
사슴	A:50, B:50	A:0, B:20	A:0, B:10
토끼	A:20, B:0	A:20, B:20	A:0, B:30
뱀	A:10, B:0	A:30, B:0	A:10, B:10

여러분은 2명씩 한 팀이 되고 사냥꾼 1명의 입장으로 대회에 참석한다.
사냥꾼들은 산에 차례대로 올라가 10일씩 지내게 된다.
예를 들어 전체 사냥꾼이 5명이면 1번 사냥꾼은 2번, 3번, 4번, 5번 사냥꾼 각각과 산속에서 10일씩 총 40일을 지내게 된다. 같은 상대방과는 무조건 2번씩만 올라가며, 리그전을 치른다고 생각해도 좋다.

첨부에서 주어진 코드(main.py, Main.java, main.cpp) 에서는 가상의 상대편 사냥꾼 3인과 경쟁을 펼치게 되며, 제출 후에는 팀간 실제 매치가 이루어진다.
이때 가장 많은 이득을 취할 수 있도록, python, c, c++언어를 선택할 경우는 Me()함수를, java언어를 선택할 경우는 hunt()메소드를 구현하여 제출한다.

[제약사항]
팀 당 한 명만 제출한다.
어떠한 외부 라이브러리나 헤더파일도 사용할 수 없다. (python, java언어를 선택한 경우 import문을 c, c++언어를 선택한 경우 #include문을 사용 시 실격임)

전역변수 사용은 가능하지만, 팀 명을 앞에 prefix로 붙인다.
예를 들어, 서울12반 2조에서 sum이라는 전역변수를 사용하고자 하면
`int seoul12_2_sum = 0;` 과 같이 선언하고,
대전, 구미, 광주의 팀에서도 sum을 사용하고자 하면 아래와 같다.
`int gwangju1_7_sum = 0;` (광주1반 7조)
`int gumi2_3_sum = 0;` (구미2반 3조)
`int daejeon4_3_sum = 0;` (대전4반 3조)
`int buulgyeong1_2_sum = 0;` (부울경1반 2조)

사슴사냥대회

[페널티]
같은 동물을 연속해서 고르게 되면(리턴하게 되면) 다음과 같은 페널티가 적용된다.

	사슴	토끼	뱀
1회 선택	0	0	0
2회 연속 선택	-1	-2	-3
3회 연속 선택	-2	-4	-6
…			
10회 연속 선택	-9	-18	-27

예를 들어 세 번 연속 토끼를 선택한 상황에서 네 번째로 다시 토끼를 선택했다면
20 – 6 = 14점을 획득하게 된다.
상대방 사냥꾼이 바뀌거나 이전 선택과 다른 동물을 선택하면 페널티 정보는
0으로 초기화된다.

[운]
매 턴마다 0,1,2 중 하나의 점수가 랜덤 하게 가산된다.

사슴사냥대회

| 구현 함수

Python : `def Me(opp, turn, opp_prev, opp_last_pattern):`

Java : `public int hunt(int opp, int turn, int opp_prev, int opp_last_pattern[][]);`

C, C++: `int Me(int opp, int turn, int opp_prev, int opp_last_pattern[][10]);`

| 인수설명

◎ **opp** : 대결할 상대방 플레이어의 번호를 나타낸다.

◎ **turn** : 0이상 9이하, 해당 플레이어와의 현재 진행 턴을 나타낸다. (총 10회)

◎ **opp_prev** : 이번 게임에서 현재 상대방의 바로 이전 선택을 나타낸다.

- -1:모름, 0:사슴, 1:토끼, 2:뱀
- turn이 0인 경우 상대방 수는 모름, turn이 1인 경우 상대방이 0번턴에 리턴한 값을 의미함
- opp_prev값이 -1, 0, 1, 2 가 아닌 값이 올 수도 있는데, 상대방이 이전 턴에 이상한 값을 리턴한 경우임

◎ **opp_last_pattern[][10]** : 현재 상대방의 바로 이전 게임 기록을 보여준다. 즉 상대방이 나와 만나기 직전 다른 사람과 게임을 했다면 그 기록을 나타낸다. 상대방 입장에서 내가 첫 상대이면 {-1,-1,...,-1}로 주어지며, 그렇지 않을 경우 0또는 1 또는 2로 채워진 배열이 주어진다. 예를 들어 {0, 1, 1, 0, 2, 2, 1, 1, 1, 0} 으로 주어진다면 {사슴, 토끼, 토끼, 사슴, 뱀, 뱀, 토끼, 토끼, 토끼, 사슴} 순으로 선택한 경우이다. (이 역시 상대방이 이상한 값을 리턴한 적이 있다면 그 정보가 반영된다)

◎ **Return** : **0를 리턴하면 현재 턴에서 사슴을 선택한다는 뜻이며, 1을 리턴하면 토끼를, 2를 리턴하면 뱀을 선택한다는 뜻.**

그 외의 값을 리턴하면 -100점의 페널티를 매 턴마다 받는다.

제공하는 첨부 소스에서 Register()함수 호출 부분을 추가하여 좀 더 많은 상대방을 넣고 테스트해보는 것도 가능하다.