컴퓨팅사고 경진대회 보고서 양식

글씨체: 맑은고딕 or Arial (한글 사용 시:함초롱바탕)

글씨크기: 10,

줄간격: 1.0 (한글 사용 시: 150%)

색다른 미니 야구 게임

창의적 게임 만들기

지원분야: 코드 구현

학과: 디지털헬스케어학과

학번: 2022247021

이름: 유민준

---------- 목차 ----------

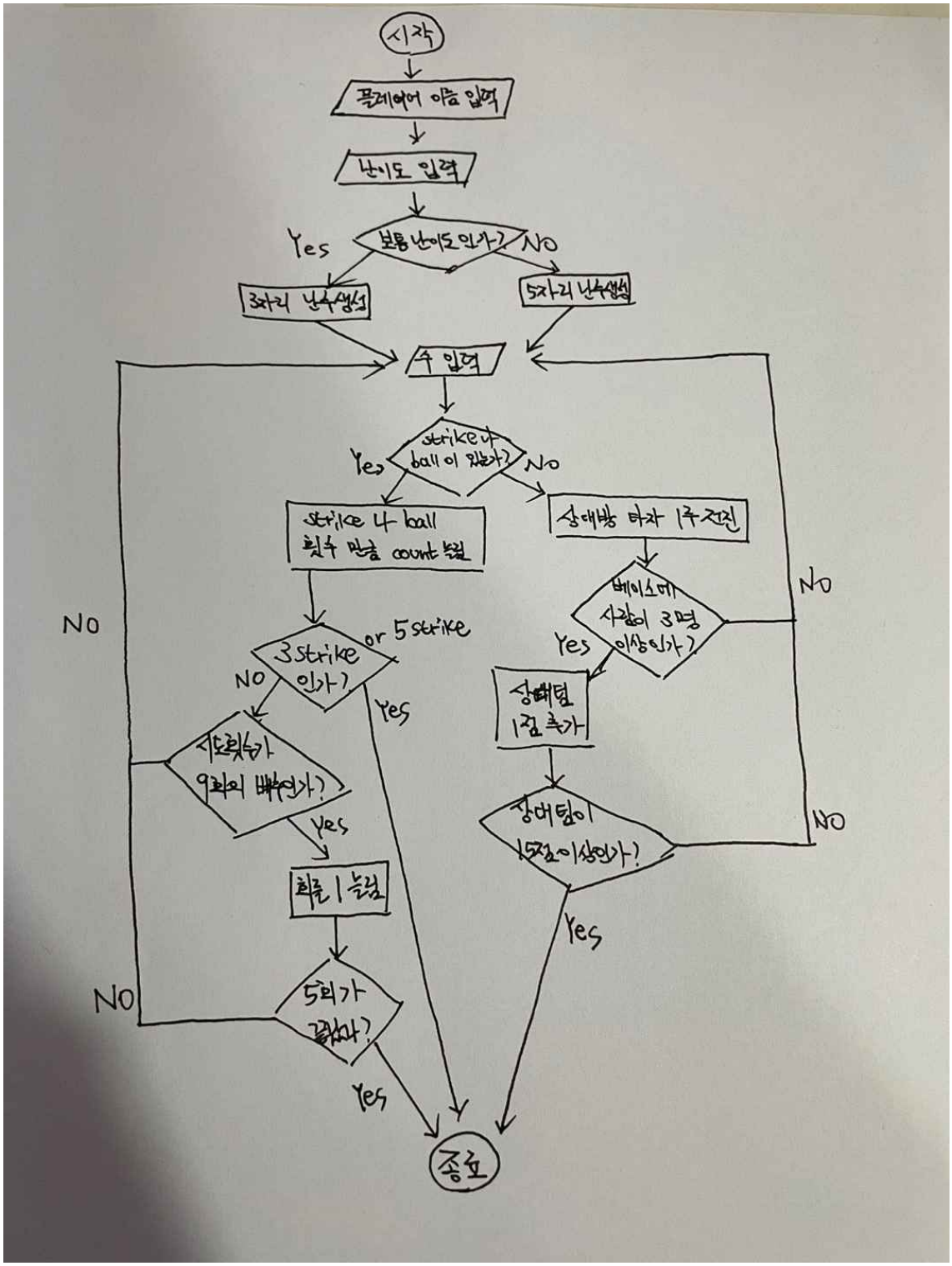
1. 저는 평소에 게임을 즐겨하는 편입니다. 그런 제가 대학교에 입학하여 처음으로 코딩을 하게 되면서, 내가 지금 두들기는 비주얼 스튜디오나 파이썬 등과 같은 코드만 적는 프로그램으로 어떻게 내가 하는 게임들을 만들 수 있을까 궁금하기도 했습니다. 물론 이제는 시중의 게임 하나를 만드는데에는 정말 여러 가지 프로그램들이 동원된다는 것을 알지만 마음 한 켠으로는 스스로 현재 많은 이들이 아는 게임이라도 짜서 구현시켜보는 것이 목표이기도 했습니다. 그래서 처음에는 컴퓨팅사고 과목을 들으며 369와 같은 기본적인 고전 게임들을 실습과 동시에 풀어보며 이러한 류의 게임을 굴리는 메커니즘이 되는 코드 짜는 패턴 등을 스스로 습득하게 되었습니다. 그리고 이번 대회를 통해 재미난 상상력을 동원하여 한 번 게임을 소소하게 나마 제가 컴퓨팅사고를 어태까지 배운 내용들을 중점적으로 하여 구현해보고자 선정하게 되었습니다.

2. 일단 우선은 우리가 평소에 많이 하는 게임들을 조사해보았습니다. 목록으로는 369, 베스킨라빈스 31 등이 있었으나 이들은 너무나도 완성형 게임이라는 생각이 들어서 선뜻 나서지 못했습니다. 그러자 머릿속에 떠오른 생각이 학창시절 수학시간에 많이하던 야구게임이었습니다. 야구게임은 실제 야구의 규칙을 일부 차용하기 때문에 기본적으로 3자리 숫자로 진행하는 것이 보통입니다. 문제 출제자는 3자리 숫자를 생각하게 되고 문제를 푸는 사람은 출제자가 생각한 3자리 숫자를 정확히 맞출 때까지 정답을 외치는 것입니다. 만약 정답자가 외친 답에서 출제자가 낸 답과 자리는 다르지만 같은 숫자가 있다면 ball, 자리와 숫자가 모두 같다면 strike가 되는 방식입니다.

3. 한계점이라고 한다면 굉장히 정형화 된 게임이라 이를 창의적으로 변형시키는 방향이 쉽지 않았습니다. 그리고 이 게임은 실제 야구의 규칙을 기반으로 하는 것인데 제가 야구에 대해서 잘 모르고 야구의 규칙이라고 한다면 실제로 뛰는 것이 아니기에 간접적으로 적용할 수 있는 규칙의 수가 적어 새로운 제안을 하는데 애를 먹었습니다.

4. 제안 목표 : 실제 야구와 야구게임을 접목시킨 게임을 만들어보자.

5. 알고리즘 설계 : 오늘 오후에 기말고사를 본 관계로 코드 구현을 하기 위해 시간상 자필로 작성하여 사진으로 올리는 점 양해 부탁드립니다...



6. 저는 기존의 야구게임에서 규칙을 많이 변환하였습니다. 특히 실제 야구의 규칙들을 많이 적용하였습니다. 기존의 야구 게임을 살펴보면 그냥 랜덤한 수를 생각하고 일방적으로 이론상 무한대의 횟수 동안 정답자가 정답을 말하며 스트라이크와 볼을 말해줍니다. 그러나 저는 이러한 횟수와 관련하여 실제 야구와 관련지어 보았습니다.

제가 설계한 것은 미니 야구게임이기 때문에 가정을 두었습니다. 먼저 게임을 플레이하는 자신은 투수의 입장입니다. 자신이 던진 공 즉 답에 따라 스트라이크와 볼이 갈립니다. 스트라이크나 볼 둘 중 하나라도 0이상이면 그냥 넘어갑니다. 그러나 0스트라이크 0볼인 경우 상대방 타자가 1루 전진합니다. 실제 야구 룰과 마찬가지로 3루까지 타자가 차있는 상태에서 안타가 나면 1득점이 나도록 했습니다. 그리고 무한정 늘릴 수 있는 정답 횟수를 조정하기 위해 '회'를 도입했습니다. 각 회당 9명의 타자가 공을 치게 되고 9명의 타자가 모두 공을 칠 시 다음 회로 넘어가는 시스템입니다. 야구에는 1회부터 9회까지 있는데 여기에서는 짧게 5회까지만 했습니다. 즉 5회를 넘어가면, 6회가 되버리면 종료가 됩니다. 그리고 몰수패 또한 도입하였습니다. 아까 말씀드렸다시피 스트라이크나 볼이 없으면 상대방이 1루 전진하고 계속되면 득점이 나오게 됩니다. (회가 늘어나도 베이스에 있는 타자는 같습니다. 회만 바뀔 뿐 진행 상황은 같음) 여기에서 상대가 15점을 기록하게 되면 플레이어는 몰수패로 지게 됩니다. 또 다른 차이점은 저는 두 가지 난이도를 설정하였습니다. 난이도는 맞춰야하는 수의 자릿수에 따라서 나뉩니다. 보통 난이도는 일반적인 3자리수, 어려움 난이도는 5자리수로 되어 있습니다.

나머지 룰은 기본적인 야구게임과 같습니다.

코드의 내용들을 참고하며 자세하게 설명하겠습니다.

먼저 가장 윗줄에 random과 radiant를 각각 선언해줍니다. 이는 컴퓨터가 생성할 3자리 혹은 5자리 난수를 위해 선언합니다.

그 다음 difficulty라는 난이도 변수를 선언하고 보통이면 1, 어려움이면 2를 입력하게 합니다.

그 다음은 크게 총 3개의 함수를 생성하였습니다. 첫 번째는 난수를 생성하는 generate_number 함수, 그 다음은 플레이어가 정답을 맞추는데 쓰는 take_guess 함수 마지막은 스트라이크와 볼 등 점수를 헤아리는 score 함수가 있습니다.

이 세가지 함수들은 일반적인 야구게임을 만드는데에도 동일하게 쓰이기 때문에 간략하게 설명하겠습니다. generate_number와 take_guess 함수는 난이도에 따라 2개씩 만들었습니다. 왜냐하면 각각 3자리와 5자리 이기 때문에 while문을 통해 정의를 할 때 3과 5를 따로 나누기 번거롭기 때문입니다. 그래서 ~~~_easy와 _hard로 나누었습니다.

generate_number 함수는 먼저 number라는 빈 리스트를 생성하고 new_number = 0을 선언하여 new_number를 radiant(0, 9)로 하여 0부터 9중 랜덤한 수를 생성하게 하고 이를 if 문에 걸어 new_number가 number 안에 없다면 append하여 new_number를 붙이게 했습니다. 그렇게 하면 new_number는 랜덤한 한 자리 숫자가 되고 이가 계속해서 숫자가 겹치지 않게 3자리 혹은 5자리 수가 만들어지는 것입니다. 그리고 number를 반환하였습니다.

take_guess 함수는 반례들을 골라내는데 중점을 두었습니다. 저는 각 자리 숫자를 따로 입력하게 설계를 하였기에 1번째 숫자에 예를 들어 9이상의 수를 쓰거나 이미 나온 수를 쓴다면 오류 메시지가 나오게 하였습니다. 그리고 이 함수 내에서도 guess와 new_guess를 만들어 위의 함수처럼 수를 하나씩 받아서 조건에 따라 골라낸 후 append하여 붙여 넣는 방식으로

하였습니다. 마지막으로 new_guess 값을 반환하였습니다.

score 함수는 간단합니다. 먼저 strike_count와 ball_count를 선언합니다. 이는 먼저 while 문에서 숫자의 길이 즉 len 만큼 조건문을 돌리는 방식으로 설계되기 때문입니다. 먼저 while i<len(guess)를 한 뒤 밑에 if-elif-else문을 걸어서 자릿수와 숫자가 모두 같다면 strike_count를 1 증가 시키고, in을 사용하여 난수 안에 해당 숫자가 포함되어 있으면, ball_count를 1 증가시키고, else면 아무 일 일어나지 않게 했습니다.

마지막 난이도 별로 게임을 시작하는 것입니다. 위의 함수 5개를 모두 선언하고 난 후 if문을 걸어 difficulty가 1이면 보통난이도, 2이면 어려움 난이도가 실행되게 하였습니다. 그 다음 함수의 반환값들을 변수로 저장하는 과정들을 거친 뒤 시도 횟수의 tries, 진행 회는 나타내는 n, 현재 베이스에 서있는 사람의 수인 stand, 상대방의 점수인 point를 선언하였습니다. 참인 동안 실행 결과의 가장 윗 줄에 회를 나타내기 위해서 회를 print하였습니다. 그리고 밑에 함수들을 각각의 매개변수들을 통해 실행해줍니다. 그 다음 strike가 3이면 종료 및 3스트라이크를 기록한 횟수도 더해줘야하기 때문에 tries도 1 증가시켰습니다. 그 다음은 회를 증가시키는 것입니다. tries를 9로 나눌 시에 나머지가 0일시 회를 1 증가시키는 것입니다. 그 다음은 strike와 ball이 모두 0일 때 안타를 기록하고 타자를 1루 전진시키는 코드입니다. 그 다음은 득점에 관한 코드인데 stand를 4라고 한 이유는 물론 5, 6이 될 수 있지만 3을 넘기자마자 1점이 오르기 때문에 이 1점은 현재 stand에서 3을 빼준 점수라고 할 수 있기 때문에 4라고 특정지었습니다. 4가 되면 point는 1점 오르고 stand는 원래 4명이지만 한 명이 홈으로 들어오기 때문에 -=1을 해주었습니다. 다음은 몰수패에 관한 코드입니다. point가 15이상이 되면 몰수패와 동시에 break가 되는 코드이고 다음은 n==6 즉 6회에 접어들었을 시에는 경기를 종료하는 코드입니다.

하단에 마지막 문단에 해당하는 코드를 첨부하였습니다.

```
if difficulty == 1:
    ANSWER = generate_numbers_easy()
    tries = 0
    n = 1
    stand = 0
    point = 0
    while 1:

        print(str(n) + "회~!!")

        GUESS = take_guess_easy()
        strike, ball = get_score(GUESS, ANSWER)
        print("{}S  {}B".format(strike, ball))

        if strike == 3:
            tries += 1
            break
        else:
            tries += 1

        if tries % 9 == 0:
            n += 1

        if strike == 0 and ball == 0:
            print(" 안타!!!...타자가 1루 전진합니다.")
            stand += 1

        if stand == 4:
            point += 1
            stand -= 1
            print(" 상대팀에서 1점 득점합니다! 현재 스코어 0 대", point ,"입니다.")

        if point >= 15:
            print("몰수패 하셨습니다...다음에 도전하세요")
            break

        if n == 6:
            print("경기가 종료되었습니다!, 최종 스코어는 0 대", point ,"입니다.")
            break

    if strike == 3:
        print("축하합니다!", str(player_name) + "님! {}번 만에 숫자 3개의 값과 위치를 모두 맞추셨습니다.".format(tries))
```

7. 결과는 제 코드가 오류 없이 구현이 되었기 때문에 성공적이라고 할 수 있습니다. 그러나 한 가지 안타까운 점은 상대방이 점수를 내는 방법은 강구를 했지만 플레이어가 점수를 내는 방법을 생각하지 못하여 조금 더 재미있게 점수를 내며 게임을 진행할 수 있었지만 3스트라이크가 되면 바로 끝이 나버리는 게임이라서 이 부분이 조금 아쉬웠습니다. 그러나 일단 무진장 길어지는 게임 시간을 줄이고 조금 더 흥미진진하게 게임이 흘러가게 만들었고, 이제 점수에 관한 부분들을 생각하여 게임의 정의를 완전히 바꿔서 다른 게임을 만들던지 하여 점수를 상대방과 서로 내어 경쟁하는 방식으로 바꾸면 훨씬 지매있어질 것이라는 기대효과가 있습니다.