로또 시뮬레이션 프로그램을 만들어 보겠습니다.

이 프로그램은 과정이 많기 때문에, 여러 파트로 나눠서 문제를 해결해 나갈 건데요. 먼저 이 레슨에서 프로그램 전체에 대한 설명을 한 번 하고 가겠습니다.

규칙

로또는 주 1회씩 열립니다. 하지만 한 사람이 한 회차에 여러 번 참여할 수도 있습니다.

번호는 1부터 45까지 있는데요. 주최측에서는 매주 6개의 '일반 당첨 번호'와 1개의 '보너스 번호'를 뽑습니다. 그리고 참가자는 1번 참여할 때마다 서로 다른 번호 6개를 선택합니다.

당첨 액수는 아래 규칙에 따라 결정됩니다.

1. 내가 뽑은 번호 6개와 일반 당첨 번호 6개 모두 일치 (10억 원)
2. 내가 뽑은 번호 5개와 일반 당첨 번호 5개 일치, 그리고 내 번호 1개와 보너스 번호 일치 (5천만 원)
3. 내가 뽑은 번호 5개와 일반 당첨 번호 5개 일치 (100만 원)
4. 내가 뽑은 번호 4개와 일반 당첨 번호 4개 일치 (5만 원)
5. 내가 뽑은 번호 3개와 일반 당첨 번호 3개 일치 (5천 원)

과제 설명

여러분의 임무는 로또 시뮬레이션을 위한 함수들을 작성하는 것입니다. 어떤 함수들이 있는지 봅시다.

generate_numbers

이 함수는 파라미터로 정수 **n**을 받습니다. 무작위로 1과 45 사이의 서로 다른 번호 **n**개를 뽑고, 그 번호들이 담긴 리스트를 리턴합니다.

예를 들어서 아래 코드를 실행하면,

```
print(generate_numbers(6))
```

이런 결과가 나올 수 있습니다.

```
[16, 2, 30, 40, 15, 33]
```

하지만 다시 실행하면 다른 결과가 나오겠죠?

참고로 이 함수는 참가자의 번호를 뽑는 데에도 쓰이고, 보너스를 포함한 당첨 번호 7개를 뽑는 데에도 쓰입니다.

draw_winning_numbers

일반 당첨 번호 6개와 보너스 번호 1개가 포함된 리스트를 리턴합니다. 일반 당첨 번호 6개는 정렬되어 있어야 하고, 보너스 번호는 마지막에 추가하면 됩니다.

예를 들어서 아래 코드를 실행하면,

```
print(draw_winning_numbers())
```

이런 결과가 나올 수 있습니다.

```
[4, 12, 14, 28, 40, 41, 6]
```

앞서 정의한 **generate_numbers** 함수를 잘 활용하면, 함수를 간결하게 작성할 수 있습니다.

count_matching_numbers

파라미터로 리스트 **list_1**과 리스트 **list_2**를 받고, 두 리스트 사이에 겹치는 번호 개수를 리턴합니다.

예를 들어서 아래 코드를 실행하면,

```
print(count_matching_numbers([2, 7, 11, 14, 25, 40], [2, 11, 13, 14, 30, 35]))
```

2, **11**, **13**이 겹치기 때문에 이렇게 나옵니다.

```
3
```

또 아래 코드를 실행하면,

```
print(count_matching_numbers([2, 7, 11, 14, 25, 40], [14]))
```

14가 겹치기 때문에 이렇게 나오겠죠?

```
1
```

check

참가자의 당첨 금액을 리턴합니다. 파라미터로 참가자가 뽑은 번호가 담긴 리스트 **numbers**와 주최측에서 뽑은 번호가 담긴 리스트 **winning_numbers**를 받는데요. **numbers**는 당연히 번호 여섯 개를 담고 있고, **winning_numbers**는 보너스까지 해서 번호 7개를 담고 있겠죠?

예를 들어서 아래 코드를 실행하면,

```
numbers_test = [2, 4, 11, 14, 25, 40]
winning_numbers_test = [4, 12, 14, 28, 40, 41, 6]

print(check(numbers_list, winning_numbers_test))
```

4, **14**, **40**... 이렇게 번호 3개가 겹치기 때문에 5천 원에 당첨됩니다.

```
5000
```