Cards

You have X black, Y white, and Z cyan cards (total $X + Y + Z$). You will play all the cards one by one onto a single pile in any order you choose.

After you place each card, check the pile:

- If the pile (now) contains at least one card of each of the three colors, you score 1 point, then all cards **except the one you just played** are destroyed (removed from the pile). The card you just played remains as the only card on the pile.
- Otherwise, nothing happens.

Your task is to calculate the maximum score you can get.

Implementation Details

You need to implement one procedure called `maximum_score`:

```
int64 maximum_score(int32 X, int32 Y, int32 Z);
```

- X: the number of black cards;
- Y: the number of white cards;
- Z: the number of cyan cards;
- This procedure might be called no more than 100 times for each test case at the beginning of the program.

The procedure should return the maximum score you can get.

Constraints

- $1 \le X \le 10^9$
- $1 \le Y \le 10^9$
- $1 \le Z \le 10^9$

Scoring

1. Subtask 1 (4 points): $X = Y = Z = 1$
2. Subtask 2 (7 points): $X + Y + Z \le 8$
3. Subtask 3 (15 points): $X + Y + Z \le 16$

4. Subtask 4 (25 points): $X, Y, Z \leq 50$
5. Subtask 5 (12 points): $X = Y = Z$
6. Subtask 6 (24 points): $X < Y = Z$
7. Subtask 7 (13 points): No additional constraints

Examples

Example 1

Consider the following call.

```
maximum_score(2, 2, 1);
```

The procedure should return 2.

Example 2

Consider the following call.

```
maximum_score(4, 3, 4);
```

The procedure should return 5.

Sample Grader

The sample grader reads the input in the following format:

- Line 1: An integer T, indicating the number of calls to `maximum_score`
- Next T lines: Three integers X, Y, and Z

The sample grader calls `maximum_score(X, Y, Z)` and prints the returned value.