

---

## Algorithmen und Wahrscheinlichkeit

### Programming Exercises 3

---

DEADLINE: 10:15 @ 6.4.2017

#### Exercise 1 – *Island tribes*

In a galaxy far, far away... there is a strange island called a BHN Island. Inhabitants of the island are divided into three tribes: *Bears*, *Hunters*, and *Ninjas* (hence the name).

The tribes are very hostile towards each other and fight for survival on this small and not too resourceful island. At every moment in time two uniformly at random inhabitants meet and if they belong to different tribes the stronger one always kills the weaker one. It is well known that *a bear is stronger than a ninja, a ninja is stronger than a hunter, and a hunter is stronger than a bear*.

After some long period of time, only members of one tribe will remain living on the island. Your task is to compute the probability of each tribe surviving.

**Input** The first line of the input file contains an integer  $1 \leq t \leq 100$  denoting the number of test cases that follow. Each of the  $t$  test cases is described as follows.

- It contains a single line with three integers  $b$   $h$   $n$ , separated by space, denoting the number of members of Bears, the number of members of Hunters, and the number of members of Ninjas, such that  $0 \leq b, h, n \leq 100$ .

**Output** For each test case output one line with the probability of each of the tribes surviving, separated by space. Each output value should be a real number rounded to seven decimal places. You should round your result with the following piece of code:

```
DecimalFormat df = new DecimalFormat("0.0#####");  
df.setRoundingMode(RoundingMode.HALF_DOWN);  
System.out.println(df.format(3.5)); // Replace 3.5 with your desired double
```

**Points** This exercise gives 1 point.

#### Sample Input

```
3  
2 2 2  
2 1 2  
1 0 0
```

#### Sample Output

```
0.3333333 0.3333333 0.3333333  
0.55 0.3 0.15  
1.0 0.0 0.0
```

## Exercise 2 – *Roulette*

You are in a casino and would like to play a game of roulette. The roulette wheel consists of  $n$  slots with, not necessarily different, numbers written on it. The rules of the game are such that a ball is placed on a slot (does not matter which one), the wheel is spun and you *win* the game if the ball lands on a slot with number 0 written on it. The wheel is *fair* in a sense that every time it is spun the ball has equal chance to land on any of the  $n$  slots.

Your task is to calculate the probability  $p$  that after spinning the wheel  $m$  times you win at least  $k$  consecutive games.

**Input** The first line of the input file contains an integer  $1 \leq t \leq 40$  denoting the number of test cases that follow. Each of the  $t$  test cases is described as follows.

- It starts with a line containing three integers  $n$   $m$   $k$ , separated by space, denoting the number of slots on the wheel, the number of times the wheel is spun, and the number of consecutive wins you are interested in, respectively, such that  $1 \leq n \leq 100, 1 \leq m \leq 100$  and  $1 \leq k \leq m$ .
- The next line contains  $n$  numbers  $v_1$   $v_2$   $\dots$   $v_n$ , separated by space, which denote the numbers written on the roulette wheel and such that  $0 \leq v_i \leq 2^{30}$  for all  $i \in \{1, \dots, n\}$ .

**Output** For each test case output one line containing the probability  $p$  rounded to 7 decimal places. You should round your result with the following piece of code:

```
DecimalFormat df = new DecimalFormat("0.0#####");  
df.setRoundingMode(RoundingMode.HALF_DOWN);  
System.out.println(df.format(3.5)); // Replace 3.5 with your desired double
```

**Points** This exercise gives 1 point.

**Hint** The problem can be solved using dynamic programming.

### Sample Input

```
5  
3 1 1  
0 0 0  
3 1 1  
1 2 3  
3 3 3  
0 1 2  
3 3 3  
0 0 1  
5 4 2  
0 1 2 3 4
```

### Sample Output

```
1.0  
0.0  
0.037037  
0.2962963  
0.104
```