Racing

Mr. Panda is designing a level for a racing game which involves racing across a series of **N** hills. He has already decided the height of the **N** hills and all of the heights will be different. However, he cannot decide on what order to place the hills to form the level. He knows the difficulty to move from a hill of height **A** to a hill of height **B** is equal to $\mathbf{(B - A)^2}$. In addition, the total difficulty of the level is the sum of the difficulty of moving from the 1st hill to the 2nd hill to the 3rd hill and so on until the **N**th hill. However, he does not want his level to be too difficult otherwise no one would play it so he wants to find out how many ways he can permute his hills such that the total difficulty is at most **D**.

Input
The input contains two integers **N** and **D** on the first line and then **N distinct** integers on the second line representing the heights of the **N** hills.

Output
The output should contain a single integer, representing the total number of ways to arrange the **N** hills such that the total difficulty of the level is at most **D**.

Limits
- $2 \le \mathbf{N} \le 9$
- $1 \le \mathbf{D} \le 10^9$
- The height of each hill will be at least 1 and at most 10000.

Sample Input (**racing1.in**)	Sample Output (**racing1.out**)
3 10 3 5 2	4

Explanation
There are 6 possible permutations of the hills and the difficulty of each level is as follows:

Permutation	**Difficulty**
2 3 5	$(3-2)^2 + (5-3)^2 = 5$
2 5 3	$(5-2)^2 + (3-5)^2 = 13$
3 2 5	$(2-3)^2 + (5-2)^2 = 10$
3 5 2	$(5-3)^2 + (2-5)^2 = 13$
5 2 3	$(2-5)^2 + (3-2)^2 = 10$
5 3 2	$(3-5)^2 + (2-3)^2 = 5$

Out of the 6 permutations, only 4 of them have difficulty at most 10 so the answer is 4.

Notes:

1. You should develop your program in the subdirectory **ex1** and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You are free to define your own helper methods and classes (or remove existing ones).
3. Please be reminded that the marking scheme is:
   a. Public Test Cases (1%) - 1% for passing **all** test cases, 0% otherwise
   b. Hidden Test Cases (1%) - Partial scoring depending on test cases passed
   c. Manual Grading (1%)
      i. Overall Correctness (correctness of algorithm, severity of bugs)
      ii. Coding Style (meaningful comments, modularity, proper indentation, meaningful method and variable names)
4. Your program will be tested with a time limit of not less than **2 sec** on Codecrunch.

Skeleton File – Racing.java

You are given the below skeleton file `Racing.java`. You should see a non-empty file when you open the skeleton file. Otherwise, you might be in the wrong working directory.

```
/**
 * Name         :
 * Matric. No   :
 * PLab Acct.   :
 */

import java.util.*;

public class Racing {
    private void run() {
        //implement your "main" method here
    }

    public static void main(String[] args) {
        Racing newRacing = new Racing();
        newRacing.run();
    }
}
```