Problem 9: Agri Yield Optimization

Time limit: 3 seconds

Imagine you are working on a project to optimize agricultural yield in a huge landscape. The landscape is used for different crops by a company in smart agricultural domain. The growth rate of each crop is represented in a static array of a fixed size, 8 i.e. ***[2,4,1,5,3,6,7,8]***. Each element in the array represents the growth rate of a specific crop at a given position and each position in the array represents the fields in the landscape. However, due to the tighter financial situation, the company wants to use the sets of contiguous fields [range of indexes, in the form of two indices; left and right] to cut down expenses.
Your task is to design an algorithm using the product of the square root to calculate the growth yield of contiguous fields. The growth yields will help the company to identify optimal contiguous regions for planting crops. One simple approach to solve this problem is to use brute force to find the overall growth yield within the given set of contiguous indexes in the array presented as a query.
However, to calculate the overall growth yield optimally in the given range you can decompose the array into components in the following way:

Number of Components = ***floor ((8 / component_size))*** where ***component_size =floor(sqrt(8))***

For keeping the accuracy, you are required to use 6 decimal places of precision for all calculations. Further, the final answer shall be represented in 3 decimal places of precision. You are also required to count the number of component hits required to reach the final answer.

Input

The first line of the input file is the number of queries ***N (1 ≤ N ≤100)***. Each subsequent line, represents the user query (values of left and right index) for your solution to calculate the potential growth of crops.

Output:

For each query, two values should be output, where first represents the potential growth and second represents the number of component hits required.

Sample input	Sample Output
3 1 6 0 7 1 5	50.2 4 200.798 4 18.974 3