/*
 * file: timeComplexity.pdf
 *
* author: Michal Tešnar <m.tesnar@student.rug.nl> S4740556
 *
 * Description: This file contains solutions of exercises to time complexity questions.
 */

1. (b)
We increase index until its square is greater or equal to the given N: i*i < n. As we assume positive integers, we can take a square root of both sides: i < √(N). Therefore we take maximum of square root of N steps.

2. (e)
In the first loop, we compute the sum of all values of i from 0 to N, which is given by formula N*(N+1)/2. The second loop resets i and decreases j until i is as big as s = N*(N+1)/2. The second loop is longer, and determines the complexity of the program to be N*N = N^2.

3. (d)
Outer loop has linear complexity (i from 0 to N). The inner loop takes squares of i and then dimishes it by dividing by 2. As we are taking the square of i, which means we have log2(N*N) = log2(N^2) = 2log2(N). We let go of constants, therefore inner loop has complexity log(N). All in all, the two loops together are O(N*log(N)).

4. (c)
While i is not equal to zero, we decrease it and it was initially set to N, so the complexity is linear.

5. (a)
The loop always diminishes the difference between two numbers by half and rounds it down (because we work with integers), until it is not greater than one. In each step, we halve the distance between the two numbers, which are initially set to be 0 and N, therefore we have log2(N) steps. It is similar to the binary search.

6. (b)
Outer loop has constant complexity, because the maximum amount of i is 5 and does not depend on the input N. The condition of the inner loop can be rewritten as j*j < N which (as we know from exercise 1) is √(N). We let go of the constant in the outer loop and presume that that change in initial j (from 1 to 5) is negligiible, therefore the whole program has complexity √(N).