An integer P is a *whole square* if it is a square of some integer Q; i.e. if $P = Q^2$.

Write a function:

```
int solution(int A, int B);
```

that, given two integers A and B, returns the number of whole squares within the interval [A..B] (both ends included).

For example, given A = 4 and B = 17, the function should return 3, because there are three squares of integers in the interval [4..17], namely $4 = 2^2$, $9 = 3^2$ and $16 = 4^2$.

Assume that:

- A and B are integers within the range [−2,147,483,648..2,147,483,647];
- $A \leq B$.

Complexity:

- expected worst-case time complexity is O(sqrt(abs(B)));
- expected worst-case space complexity is O(1).

Two non-negative integers N and M are called *similar* if their decimal representations (without leading zeros) can be obtained from each other by rearranging the digits. For example, in each of the following pairs, one integer is similar to the other:

- 123 and 312,
- 52832 and 22835,
- 12 and 12.

Write a function:

```
int solution(int N);
```

that, given a non-negative integer N, returns the number of non-negative integers similar to N.

For example, given N=1213 the function should return 12 because there are twelve integers similar to 1213, namely: 1123, 1132, 1213, 1231, 1312, 1321, 2113, 2131, 2311, 3112, 3121 and 3211.

Assume that:

- N is an integer within the range [0..99,999].

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

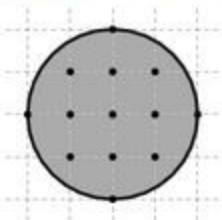
A point in a two-dimensional Cartesian plane is called a *lattice point* if both of its coordinates are integers.

Write a function:

```
int solution(int N);
```

that, given a non-negative integer N, returns the number of lattice points lying inside or on the edge of a disc of radius N which is centered at (0, 0). The function should return −1 if this number exceeds 1,000,000,000.

For example, given N = 2, the function should return 13, because there are thirteen points lying inside or on the edge of a disc of radius 2 centered at (0, 0), as indicated in the following image:



Assume that:

- N is an integer within the range [0..20,000].

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(1).

A sequence of numbers is called *arithmetic* if it consists of at least three elements and if the difference between any two consecutive elements is the same. For example, these are arithmetic sequences:

```
1,  3,  5,  7,  9
7,  7,  7,  7
3, -1, -5, -9
```

The sequence [1, 1, 2, 5, 7] is not arithmetic.

A zero-indexed array A consisting of N numbers is given. A *slice* of that array is any pair of integers (P, Q) such that $0 \leq P < Q < N$.

A slice (P, Q) of array A is called *arithmetic* if the sequence:

$$A[P], A[P+1], \ldots, A[Q-1], A[Q]$$

is arithmetic. In particular, this means that $P + 1 < Q$.

Write a function:

```
int solution(int A[], int N);
```

that, given array A consisting of N numbers, returns the number of arithmetic slices in A. The function should return −1 if the result exceeds 1,000,000,000.

For example, given array A such that:

```
A[0] = -1
A[1] =  1
A[2] =  3
A[3] =  3
A[4] =  3
A[5] =  2
A[6] =  1
A[7] =  0
```

the function should return 5 because there are five arithmetic slices of that array, namely:

```
(0, 2)  (2, 4)  (4, 6)  (4, 7)  (5, 7)
```

Assume that:

- N is an integer within the range [0..60,000];
- each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.