Question 1

**Problem Description:**

A small frog wants to get to the other side of the road. The frog is currently located at position X and wants to get to a position greater than or equal to Y. The small frog always jumps a fixed distance, D. Count the minimal number of jumps that the small frog must perform to reach its target.

Write a program that, given three integers X, Y and D, returns the minimal number of jumps from position X to a position equal to or greater than Y.

For example, given:

X = 10

Y = 85

D = 30

the function should return 3, because the frog will be positioned as follows:

after the first jump, at position 10 + 30 = 40

after the second jump, at position 10 + 30 + 30 = 70

after the third jump, at position 10 + 30 + 30 + 30 = 100

Write an efficient algorithm for the following assumptions:

X, Y and D are integers within the range [1..1,000,000,000]; X ≤ Y.

Question 2

**Problem Description:**

You are playing a counting game with your friend. Your friend will give you a string like “1020030004002920”, you need to count out the maximal sequence of consecutive zeros, for this example, the result is 3.

**Input Specification**

input a string only including number 0~9, its length is in between 1 and 100.

**Output Specification**

The output will be an integer D, which is the maximal sequence of consecutive zeros.

**Sample Input1**:

11000000020000000000000000

**Output for Sample Input1:**

7

**Sample Input2**:

000000201

**Output for Sample Input**

1

Question 3

**Problem Description:**

Your friend give you a number ‘5’, and another number. you must use these two numbers to build a possible maximum number via inserting ‘5’ to any digit position of number two.

**Input Specification**

The first line of input consists of an integer T (-10000 ≤ T ≤ 10000), which is the number two.

**Output Specification**

The output will be the maximum number built by the inputted number and 5.

**Sample Input 1**

347

**Output for Sample Input 1**

5347

**Sample Input 2**

-708

**Output for Sample Input 2**

-5708

Question 4

**Problem Description:**

A small frog wants to get to the other side of a river. The frog is initially located on one bank of the river (position 0) and wants to get to the opposite bank (position X+1). Leaves fall from a tree onto the surface of the river.

You are given an array A consisting of N integers representing the falling leaves. A[K] represents the position where one leaf falls at time K, measured in seconds.

The goal is to find the earliest time when the frog can jump to the other side of the river. The frog can cross only when leaves appear at every position across the river from 1 to X (that is, we want to find the earliest moment when all the positions from 1 to X are covered by leaves). You may assume that the speed of the current in the river is negligibly small, i.e. the leaves do not change their positions once they fall in the river.

For example, you are given integer X = 5 and array A such that:

A[0] = 1

A[1] = 3

A[2] = 1

A[3] = 4

A[4] = 2

A[5] = 3

A[6] = 5

A[7] = 4

In second 6, a leaf falls into position 5. This is the earliest time when leaves appear in every position across the river.

Write a program that, given a non-empty array A consisting of N integers and integer X, returns the earliest time when the frog can jump to the other side of the river.

If the frog is never able to jump to the other side of the river, the function should return -1.

For example, given X = 5 and array A such that:

A[0] = 1

A[1] = 3

A[2] = 1

A[3] = 4

A[4] = 2

A[5] = 3

A[6] = 5

A[7] = 4

the function should return 6, as explained above.

Write an efficient algorithm for the following assumptions:

N and X are integers within the range [1..100,000]; each element of array A is an integer within the range [1..X].

Question 5

**Problem Description:**

You are given N counters, initially set to 0, and you have two possible operations on them:

increase(X) ? counter X is increased by 1,

max counter ? all counters are set to the maximum value of any counter.

A non-empty array A of M integers is given. This array represents consecutive operations:

if A[K] = X, such that 1 ≤ X ≤ N, then operation K is increase(X),

if A[K] = N + 1 then operation K is max counter.

For example, given integer N = 5 and array A such that:

A[0] = 3

A[1] = 4

A[2] = 4

A[3] = 6

A[4] = 1

A[5] = 4

A[6] = 4

the values of the counters after each consecutive operation will be:

(0, 0, 1, 0, 0)

(0, 0, 1, 1, 0)

(0, 0, 1, 2, 0)

(2, 2, 2, 2, 2)

(3, 2, 2, 2, 2)

(3, 2, 2, 3, 2)

(3, 2, 2, 4, 2)

The goal is to calculate the value of every counter after all operations.

Write a program that, given an integer N and a non-empty array A consisting of M integers, returns a sequence of integers representing the values of the counters.

Result array should be returned as an array of integers.

For example, given:

A[0] = 3

A[1] = 4

A[2] = 4

A[3] = 6

A[4] = 1

A[5] = 4

A[6] = 4

the function should return [3, 2, 2, 4, 2], as explained above.

Write an efficient algorithm for the following assumptions:

N and M are integers within the range [1..100,000];

each element of array A is an integer within the range [1..N + 1].

Question 6

**Problem Description:**

There is a queue of N cars waiting at a filling station. there are three fuel dispensers at the station, labeled X, Y and Z, respectively. Each dispenser has some finite amount of fuel in it; at all times the amount of available fuel is clearly displayed on each dispenser.

When a car arrives at the front of the queue, the driver can choose to drive to any dispenser not occupied by another car. Suppose that the fuel demand is D liters for this car. The driver must choose a dispenser which has at least D liters of fuel. If all unoccupied dispensers have less than D liters, the driver must wait for some other car to finish tanking up. If all dispensers are unoccupied, and no has at least D liters, the driver is unable to refuel the car and it blocks the queue indefinitely. If more than one unoccupied dispenser has at least D liters, the driver chooses the one labeled with the smallest letter among them.

Each driver will have to wait some amount of time before he or she starts refueling the car. Calculate the maximum waiting time among all drivers. Assume that tanking

one liter of fuel takes exactly one second, and moving cars is instantaneous.

Write a program that, given an array A consisting of N integers (which specify the fuel demandmands in liters for subsequent cars in the queue), and numbers X, Y and Z (which specify the initial amount of fuel in the respective dispensers), returns the maximum waiting time for a car . if any car is unable to refuel, the function should return -1.

For example, given X=7, Y=11, Z=3 and the following array A is:

A[0] = 2

A[1] = 8

A[2] = 4

A[3] = 3

A[4] = 2

the program should return 8. The subsequent cars will have to wait in the queue for 0, 0, 2, 2 and 8 seconds, respectively. The scenario is as follows:

At time 0, car 0 drives to dispenser X.

At time 0, car 1 drives to dispenser Y.

There is no enough fuel in dispenser Z to satisfy the demands of car 2, so this car must wait. At time 2, car 0 finishes refueling and car 2 drives to dispenser X.

At time 2 car 3 drives to dispenser Z.

At this time all dispensers are occupied, so car 4 waits. There will be not enough fuel in dispensers X and Z after car 2 and car 3 finish tanking up, so

car 4 waits until car 1 finishes refuelling ar dispenser Y. At time 8, car 4 drives to dispenser Y.