**LEBANESE AMERICAN UNIVERSITY**

**Department of Computer Science and Mathematics**

**CSC 310: Algorithms and Data Structures**

Fall 2013

**Lab VII**

**Dynamic Programming**

**Input:**

All inputs are read from a file labeled “**n**.in” were **n** is the problem number. In the file, You read an integer **k** the number of test cases then **k test** cases follow.

**Output:**

Output should be consistent with the output specified in each problem as specified.

**Problem 1**

The old song declares "Go ahead and hate your neighbor", and the residents of One Tinville have taken those words to heart. Every resident hates his next-door neighbors on both sides. Nobody is willing to live farther away from the town's well than his neighbors, so the town has been arranged in a big circle around the well. Unfortunately, the town's well is in disrepair and needs to be restored. You have been hired to collect donations for the Save Our Well fund.

Each of the town's residents is willing to donate a certain amount which is listed in clockwise order around the well. However, nobody is willing to contribute to a fund to which his neighbor has also contributed. Next-door neighbors are always listed consecutively in **donations**, except that the first and last entries in **donations** are also for next-door neighbors. You must calculate and print the maximum amount of donations that can be collected.

**Sample Input: Sample Output:**

**3** 19

**6** 10 3 2 5 7 8 16

**10** 1 2 3 4 5 1 2 3 4 5

**Problem 2**

Given a string of digits, find the minimum number of additions required for the string to equal some target number. Each addition is the equivalent of inserting a plus sign somewhere into the string of digits. After all plus signs are inserted, evaluate the sum as usual. For example, consider the string "12" (quotes for clarity). With zero additions, we can achieve the number 12. If we insert one plus sign into the string, we get "1+2", which evaluates to 3. So, in that case, given "12", a minimum of 1 addition is required to get the number 3. As another example, consider "303" and a target sum of 6. The best strategy is not "3+0+3", but "3+03". You can do this because leading zeros do not change the result. The first input is a **String** and the second input is an **Integer**.

You must print the minimum number of additions required to create an expression from **numbers** that evaluates the sum. If this is impossible, print -1.

**Sample Input: Sample Output:**

**2** 4

99999 45 3

1110 3

**Problem 3**

A table composed of **N x M** cells, each having a certain quantity of apples, is given. You start from the upper-left corner. At each step you can go down or right one cell. Find the maximum number of apples you can collect.

**Sample Input: Sample Output:**

**3** 192

**2 7** 170

7 12 29 27 31 34 30

18 5 21 18 32 9 22

**2 7**

28 24 12 27 7 5 23

6 7 20 14 6 30 29

**Problem 4**

Given a sequence of n real numbers A(1) ... A(n), determine a contiguous subsequence A(i) ... A(j) for which the sum of elements in the subsequence is maximized.

Find the sum of the subsequence.

**Sample Input: Sample Output:**

**2** 107

**6** -76 -79 19 27 61 -69 167

**4** 98 16 53 -28

**Problem 5**

You are planting a flower garden with bulbs to give you joyous flowers throughout the year. However, you wish to plant the flowers such that they do not block other flowers while they are visible.

You will be given a int[] height, a int[] bloom, and a int[] wilt. Each type of flower is represented by the element at the same index of height, bloom, and wilt. height represents how high each type of flower grows, bloom represents the morning that each type of flower springs from the ground, and wilt represents the evening that each type of flower shrivels up and dies. Each element in bloom and wilt will be a number between 1 and 365 inclusive, and wilt[*i*] will always be greater than bloom[*i*]. You must plant all of the flowers of the same type in a single row for appearance, and you also want to have the tallest flowers as far forward as possible. However, if a flower type is taller than another type, and both types can be out of the ground at the same time, the shorter flower must be planted in front of the taller flower to prevent blocking. A flower blooms in the morning, and wilts in the evening, so even if one flower is blooming on the same day another flower is wilting, one can block the other.

You should return a int[] which contains the elements of height in the order you should plant your flowers to achieve the above goals. The front of the garden is represented by the first element in your return value, and is where you view the garden from. The elements of height will all be unique, so there will always be a well-defined ordering.

**Sample Input: Sample Output:**

**3** 1 2 3 4 5

**5** 3 4 5 1 2

5 4 3 2 1 4 5 2 3

1 1 1 1 1

365 365 365 365 365

**5**

5 4 3 2 1

1 5 10 15 20

5 10 15 20 25

**4**

3 2 5 4

1 2 11 10

4 3 12 13

**Problem 6**

Farmer John's arch-nemesis, Farmer Paul, has decided to sabotage Farmer John's milking equipment!

The milking equipment consists of a row of N (3 <= N <= 100,000) milking machines, where the ith machine produces Mi units of milk (1<= Mi <= 10,000). Farmer Paul plans to disconnect a contiguous block of these machines -- from the ith machine up to the jth machine (2 <=i <= j <= N-1); note that Farmer Paul does not want to disconnect either the first or the last machine, since this will make his plot too easy to discover. Farmer Paul's goal is to minimize the average milk production of the remaining machines. Farmer Paul plans to remove at least 1 cow, even if it would be better for him to avoid sabotage entirely.

Fortunately, Farmer John has learned of Farmer Paul's evil plot, and he is wondering how bad his milk production will suffer if the plot succeeds. Please help Farmer John figure out the minimum average milk production of the remaining machines if Farmer Paul does succeed.

**Sample Input: Sample Output:**

1 2.667

5

5

5

7

8

2

## Problem 7

A salesman in the wild west. He was travelling from a small town S to another small town T, but did not want to pay more for the transport than necessary. The means of transportation was stagecoaches, that were scheduled between the small towns in the area. The cost of travel was to be found by adding together the cost of the individual distances. Given a schematic map of the possible routes in the form of the graph below with the costs of the individual stretches associated with the arcs, the problem can be formulated as finding the total cost of the cheapest path from S to T in the graph.