CONNECT WITH INNOVATION

# SOFTEC '18

# March 10-11, 2018

www.softecnu.org

# Sample Problems Belal Hashmi Programming Competition

## PROBLEM 1

### PropBot

You have been selected to write the navigation module for PropBot. Unfortunately, the mechanical engineers have not provided a lot of flexibility in movement; indeed, the PropBot can only make two distinct movements. It can either move 10 cm forward, or turn towards the right by 45 degrees. Each of these individual movements takes one second of time.

**Input**

Your module has two inputs: the Cartesian coordinates of a point on the plane that the PropBot wants to get as close to as possible, and the maximum number of seconds that can be used to do this. At the beginning of the navigation, the robot is located at the origin, pointed in the +x direction. The number of seconds will be an integer between 0 and 24, inclusive. Both the x and y coordinates of the desired destination point will be a real number between -100 and 100, inclusive. The first entry in the input file will be the number of test cases, t ($0 < t \leq 100$). Following this line will be t lines, with each line containing three entries separated by spaces. The first entry will be the number of seconds PropBot has to get close to the point. The second entry is the x-coordinate of the point, and the third entry is the y-coordinate of the point.

**Output**

Your program must return the distance between the goal point and the closest point the robot can get to within the given time. Your result should include at least one digit to the left of the decimal point, and exactly six digits to the right of the decimal point. To eliminate the chance of round off error affecting the results, we have constructed the test data so the seventh digit to the right of the decimal point of the true result is never a 4 or a 5.

**Sample Input**

```
2
24 5.0 5.0
9 7.0 17.0
```

**Sample Output**

```
0.502525
0.100505
```

# PROBLEM 2

## Pencils from the 19th Century

Before "automaton" was a theoretic computer science concept, it meant "mechanical figure or contrivance constructed to act as if by its own motive power; robot." Examples include fortunetellers, as shown above, but could also describe a pencil seller, moving pencils from several baskets to a delivery trough. On National Public Radio, the Sunday Weekend Edition program has a "Sunday Puzzle" segment. The show that aired on Sunday, 29 June 2008, had the following puzzle for listeners to respond to (by Thursday, 3 July, at noon through the NPR web site): From a 19th century trade card advertising Bassetts Horehound Troches, a remedy for coughs and colds: A man buys 20 pencils for 20 cents and gets three kinds of pencils in return. Some of the pencils cost four cents each, some are two for a penny and the rest are four for a penny. How many pencils of each type does the man get? One clarification from the program of 6 July: correct solutions contain at least one of each pencil type. For our purposes, we will expand on the problem, rather than just getting 20 pencils for 20 cents (which is shown in the sample output below). The input file will present a number of cases. For each case, give all solutions or print the text "No solution found". Solutions are to be ordered by increasing numbers of four-cent pencils.

### Input

Each line gives a value for N ($2 < N < 256$), and your program is to end when N=0 (at most 32 problems).

### Output

The first line gives the instance, starting from 1, followed by a line giving the statement of the problem. Solutions are shown in the three-line format below followed by a blank line, or the single line "No solution found", followed by a blank line. Note that by the nature of the problem, once the number of four-cent pencils is determined, the numbers of half-cent and quarter-cent pencils are also determined.

```
Case n:
nn pencils for nn cents
nn at four cents each
nn at two for a penny
nn at four for a penny
```

### Sample Input

```
10
20
40
0
```

### Sample Output

```
Case 1:
```

```
10 pencils for 10 cents
No solution found.

Case 2:
20 pencils for 20 cents
3 at four cents each
15 at two for a penny
2 at four for a penny

Case 3:
40 pencils for 40 cents
6 at four cents each
30 at two for a penny
4 at four for a penny

7 at four cents each
15 at two for a penny
18 at four for a penny
```

## PROBLEM 3

### Obstacle Course

You are working on the team assisting with programming for the Mars rover. To conserve energy, the rover needs to find optimal paths across the rugged terrain to get from its starting location to its final location. The following is the first approximation for the problem. N x N square matrices contain the expenses for traversing each individual cell. For each of them, your task is to find the minimum-cost traversal from the top left cell [0][0] to the bottom right cell [N-1][N-1]. Legal moves are up, down, left, and right; that is, either the row index changes by one or the column index changes by one, but not both.

**Input**

Each problem is specified by a single integer between 2 and 125 giving the number of rows and columns in the N x N square matrix. The file is terminated by the case N = 0. Following the specification of N you will find N lines, each containing N numbers. These numbers will be given as single digits, zero through nine, separated by single blanks.

**Output**

Each problem set will be numbered (beginning at one) and will generate a single line giving the problem set and the expense of the minimum-cost path from the top left to the bottom right corner, exactly as shown in the sample output (with only a single space after "Problem" and after the colon).

**Sample Input**

```
3
5 5 4
```

```
3 9 1
3 2 7
5
3 7 2 0 1
2 8 0 9 1
1 2 1 8 1
9 8 9 2 0
3 6 5 1 5
7
9 0 5 1 1 5 3
4 1 2 1 6 5 3
0 7 6 1 6 8 5
1 1 7 8 3 2 3
9 4 0 7 6 4 1
5 8 3 2 4 8 3
7 4 8 4 8 3 4
0
```

**Sample Output**

```
Problem 1: 20
Problem 2: 19
Problem 3: 36
```

# PROBLEM 4

## Assignments

When Starfleet headquarters gets a request for an exploration expedition, they need to determine which ship from those currently docked in the docking bay to send. They decide to send whichever ship is currently able to make the expedition based on how much fuel is currently stored on the ship as well as how long it will take the ship to arrive at the expected destination. Due to the age and current maintenance of the ships, each ship travels at a different top speed and has a different fuel consumption rate. Each ship reaches its top speed instantaneously.

### Input

Input begins with a line with one integer T (1 ≤ T ≤ 50) denoting the number of test cases. Each test case begins with a line with two space-separated integers N and D, where N (1 ≤ N ≤ 100) denotes the number of ships in the docking bay and D (1 ≤ D ≤ 106 ) denotes the distance in lightyears to the expedition site. Next follow N lines with three space-separated integers vi , fi , and ci , where vi (1 ≤ vi ≤ 1000) denotes the top speed of ship i in light-years per hour, fi (1 ≤ fi ≤ 1000) denotes the fuel on ship i in kilos of deuterium, and ci (1 ≤ ci ≤ 1000) denotes the fuel consumption of ship i in kilos of deuterium per hour.

**Output**

For each test case, print a single integer on its own line denoting the number of ships capable of reaching the expedition site. Be careful with integer division!

**Sample Input**

```
2
3 100
52 75 10
88 13 44
56 9 5
2 920368
950 950 1
943 976 1
```

**Sample Output**

```
2
1
```

# PROBLEM 5

## Federation Favorites

En route to Rigel 7, Chief Engineer Geordi Laforge and Data were discussing favorite numbers. Geordi exclaimed he preferred Narcissistic Numbers: those numbers whose value is the same as the sum of the digits of that number, where each digit is raised to the power of the number of digits in the number. Data agreed that Narcissistic Numbers were interesting, but not as good as his favorite: Perfect Numbers. Geordi had never heard of a Perfect Number, so Data elaborated, “A positive integer is said to be Perfect if it is equal to the sum of its positive divisors less than itself. For example, 6 is Perfect because 6 = 1 + 2 + 3.” Geordi began thinking about an algorithm to determine if a number was Perfect, but did not have the raw computing ability of Data. He needs a program to determine if a given number is Perfect. Help Geordi write that program

**Input**

Input consists of a single entry per line. Each line contains a single positive integer n, where 2 < n < 100, 000 for each case. A line containing -1 denotes the end of input and should not be processed.

**Output**

For each case, determine whether or not the number is Perfect. If the number is Perfect, display the sum of its positive divisors less than itself. The ordering of the terms of the sum must be in ascending order. If a number is not Perfect, print “ is NOT perfect.” where is the number in question. There must be a single space between any words, symbols, or numbers in all output, with the exception of the period at the end of the sentence when a number is not perfect.

**Sample Input**

```
6
12
28
-1
```

**Sample Output**

```
6 = 1 + 2 + 3
12 is NOT perfect.
28 = 1 + 2 + 4 + 7 + 14
```