18th Annual Denison Spring Programming Contest
24 February 2007

Rules:

1. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
2. All programs will be re-compiled prior to testing with the judges' data.
3. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed.
4. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
5. All communication with the judges will be handled by the PC^2 environment.
6. The allowed programming languages are C, C++ and Java.
7. Judges' decisions are to be considered final.
8. There are **six** questions to be completed in **four hours**.
9. **Important:** No extraneous whitespace should appear in your output. Specifically, there should be no blank lines, unless specifically called for. A line should *never* end with whitespace. Do not use tabs in your output, only spaces. Data fields in an output line should be separated by a single whitespace, unless specified otherwise. Any deviations to these guidelines will result in a "Format Error" from the judges.

Problem A: Sum the Dice

Suppose a bunch of dice (values 1 through 6) are rolled and arranged in an array. See if you can find three contiguous dice whose sum is a given number. Dice are contiguous (touching) if one of their sides is touching. So, three contiguous dice would be three in a row, three in a column or three in an L-shape.

For example, the following array of dice has 8 appear as the sum of 3 dice in 8 ways. (Some overlap.)

2	3	1	4
6	6	6	5
1	1	2	3
2	2	5	5
1	1	1	1

Input

Each input set will start with a row of three integers r s n (with $2 \leq r, s, \leq 10$). A value of $n = 0$ indicates end of input. The next line will contain rs integers between 1 and 6, giving the r by s array of dice with the first row of s dice, left-to-right, then the second row, through to the rth row.

Output

Each input set should produce a line of the form.

`Array` a `has the sum` n `appear` k `times.`

where a is the input set number (starting at 1), and k is the number of times you find n being the sum of 3 contiguous dice.

Sample Input

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

Sample Output

```
Array 1 has the sum 8 appear 8 times.
```

Problem B: Sweet Tarts

Sweet Tarts are a hard candy similar to Life Savers. They come in a roll with n Sweet Tarts per roll and there are four flavors (A, B, C, and D). The package is clear so you can see the order of the flavors in the roll. You can only get a Sweet Tart from either end of the roll; no breaking open the package in the middle. Furthermore, you immediately eat a Sweet Tart once you remove it from the package. Now you have a definite preference for the flavors and like to save your favorites until the end. Being the computer science type, you've come up with a score for the order in which you eat the candies. You give each of the flavors a number from 1 through 4, depending on your preference (4 is most favored, 1 is least). The score for eating a Sweet Tart is the product of its preference number with the position you ate it. For example if you ate a flavor B as the 5th Sweet Tart, and B had preference number 3, then it's score would be 15.

After you've eaten the entire roll, the score is the sum of all the individual Sweet Tarts eaten. Of course, you'd like to maximize the score. Given a particular roll of Sweet Tarts and the preference order of the four flavors, you'll compute the maximum score of the roll. (Note there may be more than one way of realizing the score.)

Input

There will be multiple input sets. Each input set will have 3 lines. The first line will contain n ($n \leq 100$), the number of Sweet Tarts in the roll. A value of $n = 0$ indicates there are no more input sets for this problem. The second line will contain a permutation of $\{A, B, C, D\}$ indicating preference for the flavors (most favored to least). The third line will contain the n flavors as they appear in the roll, from left to right. Note that it may be that not all flavors will be used.

Output

Each input set should produce one line of output of the form:

`Case` i `has a maximum score of` s`.`

where i is the number of the input set (starting at 1) and s is the maximum score you determine.

Sample Input

```
4
BCDA
ABCD
5
BACD
AAAAA
0
```

Sample Output

```
Case 1 has a maximum score of 30.
Case 2 has a maximum score of 45.
```

Problem C: The Taxman

Taxman is a one-person game played as follows: The game starts with the set of integers $S = \{1, 2, \ldots, n\}$ and the player picks any number in S with a proper divisor in S. The player gets the number picked and the taxman gets all the proper divisors in S. The number and its proper divisors are then removed from S. Play continues until the only numbers remaining have no proper divisors in S. The taxman then gets all the numbers left. You win the game if the sum of your numbers is greater than the sum of the taxman's numbers. The *score* of the game is the taxman's sum subtracted from your sum.

You are going to test some strategies in playing taxman. We'll call any number in S with proper divisors in S a *live* number.

Strategy A: Pick the largest live number with divisors.

Strategy B: Pick the live number whose sum of divisors left is the smallest. (If a tie, pick largest such live number.)

Strategy C: Pick the live number with the fewest divisors. (If a tie, pick largest such live number.)

Input

There will be multiple input sets for this problem. Input for each input set will be a single integer n ($2 \leq n \leq 100$). $n = 0$ indicates end of input.

Output

Each input set will produce one output line of the form:

`Input` k`:` A B C

where k is the number of the input set (starting at 1) and A, B and C are the scores of the game using strategies A, B, and C, respectively.

Sample Input

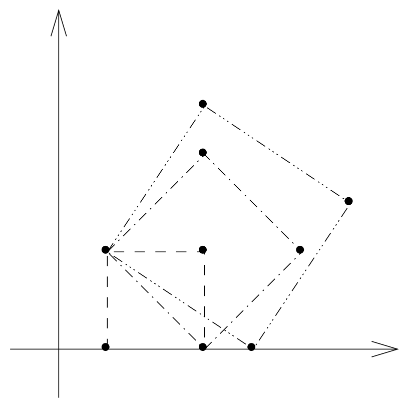
```
10
100
5
0
```

Sample Output

```
Input 1: -1 5 25
Input 2: -932 620 890
Input 3: 3 3 3
```

Problem D: Squares

Consider the dots in the diagram below, which are all on lattice points (that is, points with integer coordinates). The diagram also shows all the squares that can be created with vertices on four dots from those given.



The problem here is to determine how many squares can be made with only the given dots as vertices.

Input

Each input set will consist of two lines. The first contains the integer n ($n \leq 40$). A value of $n = 0$ indicates end of input. The next line will contain n pairs of integers (both numbers ≥ 0 and ≤ 40), each representing the coordinates of a distinct dot.

Output

Each input set should produce a line of the form

`Case` k `has` s `squares.`

where k is the input set number (starting at 1) and s is the number of squares possible, as described above.

Sample Input

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

Sample Output

```
Case 1 has 3 squares.
```

Problem E: Packing Boxes

Bob, of Bob's Big Box Barn, has many sizes of boxes. All are the usual rectangular boxes but various dimensions. Bob ships samples of these boxes for display purposes, and likes to ship as many as he can but only wishes to ship them all in one box, to keep expenses down.

When packing boxes, he starts with the smallest box and puts it entirely inside a larger one, closing the flaps of the larger one. Then he takes that box and puts it inside another, closing the flaps, and so on. The last box is then closed and sealed and shipped. Bob wants to know the largest number of *different sized* boxes he can pack this way. Note that Bob can set the inner box on any side inside the larger box. So box A might not fit inside box B in one orientation, but would in another. For example, a 3 by 4 by 5 box would fit inside a 5 by 6 by 4 box, but not in all orientations. To simplify things, we'll ignore thickness of the box sides. So, a 3 by 4 by 5 box will fit inside of another 3 by 4 by 5 box (although two boxes of the same dimension will not occur for us).

For example, if Bob has boxes of size 3 by 4 by 5, 5 by 6 by 3, 1 by 4 by 10, 4 by 4 by 4 and 6 by 6 by 5, he could pack 3 in one box. This could be done by putting 3 by 4 by 5 in a 5 by 6 by 3 in a 6 by 6 by 5. (There could be more than one way of doing this.)

Input

There will be multiple input sets for this problem. Input for each set will start with one line containing a single integer n ($n \leq 50$). $n = 0$ indicates end of input. There will then follow n lines, each with three positive integers (all no greater than 100), representing the dimensions of a box. You may assume no two boxes will have the same dimensions.

Output

Each input set will produce one output line of the form:

`Set` k can pack b boxes.

where k is the number of the input set (starting at 1) and you compute b.

Sample Input

```
5
3 4 5
5 6 3
1 4 10
4 4 4
6 6 5
3
3 1 7
5 1 4
2 5 3
0
```

Sample Output

```
Set 1 can pack 3 boxes.
Set 2 can pack 1 boxes.
```

Problem F: Square Digits

What's the pattern in the following series: $4, 16, 37, 58, 89, 145, 42, 20, 4, 16, \cdots$? (Don't spend contest time thinking about the answer...we're about to tell you!). Each term is the sum of the squares of the digits of the previous term ($16 = 4^2$, $37 = 1^2 + 6^2$, $58 = 3^2 + 7^2$, etc.). As you can see, the series above repeats itself, with a cycle of length 8. It will always happen that this series will eventually repeat a number. The number of numbers in this repetitive loop is the *cycle length*. For this problem, you will be given a starting integer and must determine the cycle length of the resulting series.

Input

Input will consist of multiple test cases. Each case will consist of a single line containing the starting integer n for a series (a value of 0 will indicate end of input). The maximum value of n will be 10000000.

Output

Each input set will produce one output line of the form:

`Set` k `has a cycle length of` c.

where k is the number of the input set (starting at 1) and you compute c.

Sample Input

```
4
1
0
```

Sample Output

```
Set 1 has a cycle length of 8.
Set 2 has a cycle length of 1.
```