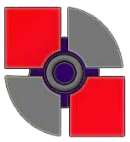
UTM
UNIVERSITI TEKNOLOGI MALAYSIA



PERSAKA
PERSATUAN MAHASISWA SAINS KOMPUTER

PROGRAMMING BATTLE

GROUP QUESTIONS

DATE OF BATTLE

10^{th} DECEMBER 2020

GROUP

1. Bubble sort is the fundamental sorting technique that is done by swapping certain pairs of adjacent entries in the array. We will obtain a **swap map** if we list the identities and sequence of the pairs to be swapped. For instance, let's say we wish to sort the array P whose elements are 5, 4, and 2 in ascending order. If the subscripts for this array are 0, 1, and 2, sorting the array can be accomplished by swapping P0 and P2, then swapping P0 and P1, and finally swapping P1 and P2. If a pair is identified in a swap map by indicating the subscript of the **first element** of the pair to be swapped, then this sorting process would be characterized with the swap map 1 0 1.

   It is useful to note that there may be several ways to do the swapping of adjacent array entries for sorting purpose. The previous array, containing 5 4 2, could also be sorted by swapping P0 and P1, then swapping P1 and P2, and finally swapping P0 and P1 again. The swap map that describes this sorting sequence is 0 1 0.

   For a given array, how many different swap maps exist? There will be an infinite number of swap maps because sequential swapping of an arbitrary pair of elements will not change the order of the elements. Thus the swap map 0 0 0 1 0 will also arrange our array elements in ascending order. But how many swap maps of minimum size will place a given array in order? That is the question you are to tackle in this problem.

   The input data will contain an arbitrary number of test cases, followed by a single 0 to tell the program to end. Each test case will have an integer n that gives the size of an array, and will be followed by the n integer values in the array. For each test case, print a message similar to those shown in the sample output below. In no test case will n be larger than 5**.**

   Sample Output:

```
2 9 7
There are 1 swap maps for input data set 1.

2 14 50
There are 0 swap maps for input data set 2.

3 3 2 1
There are 2 swap maps for input data set 3.

3 5 1 9
There are 1 swap maps for input data set 4.

0
```

2. There is a story about a chimpanzee and five men. They were shipwrecked on an isolated island. On the first night, they worked together to collect coconuts at the island. During the night, one man could not fall asleep and decided to take his share of the coconuts. Then, he divided them into five piles. There was one coconut left over so he gave it to the chimpanzee. Afterwards, he hid his share and went back to sleep.

   Soon a second man woke up and did the same thing. Once again, there's one coconut left after dividing the coconuts into five piles. He then gave the extra one coconut to the chimpanzee kindly. The third, fourth, and fifth man followed precisely the same procedure. The next morning, after waking up, they divided the remaining coconuts into five equal shares. However, there's no coconuts were left over.

   An obvious question is "how many coconuts did they originally gather?" There are an infinite number of answers, but the lowest of these is 3,121. But that's not our problem here.

   We turn the problem around. Suppose we know the number of coconuts that were collected, then what is the maximum number of persons (and one chimpanzee) that could have been shipwrecked if the same procedure could occur?

   Input

   The input will consist of a sequence of integers, each representing the number of coconuts collected by a group of persons (and a chimpanzee) that were shipwrecked. The sequence will be followed by a negative number to end the algorithm.

   Output

   Determine the largest number of persons who could have taken part in the procedure described above for each input coconut number. Display the results similar to the manner shown below, in the Sample Output. There may be no solution for some of the input cases; if so, state that no solution.

   Sample Output:

```
25
25 coconuts, 3 people and 1 monkey

3120
3120 coconuts, no solution

35
35 coconuts, 2 people and 1 monkey

99
99 coconuts, 2 people and 1 monkey

44
44 coconuts, no solution

-1
```

3. Bowling is a fun activity to do. There are ten frames in a single bowling game. The player needs to roll a ball in each frame towards ten bowling pins, which are arranged in an equilateral triangle. He or she will try to knock down as many pins as possible.

   For each frame, the bowler is allowed a maximum of two attempts to knock down all ten pins. If the bowler knocks them all down on the first attempt, the frame is scored as a **strike**. Otherwise, the bowler is allowed a second attempt to knock down the remaining pins. If the bowler succeeds in knocking the rest of the pins down in the second attempt, the frame is scored as a **spare**.

   The score for a bowling game consists of the total of the scores for each frame. The score for each frame is the total number of pins knocked down in the frame, plus bonuses for strikes and spares. Specifically, if a bowler scores a **strike** in a particular frame, the **score for that frame is ten plus the sum of the next two rolls**. If a bowler scores a **spare** in a particular frame, the score for that frame is **ten plus the score of the next roll**. If a bowler scores a **strike** in the **tenth (final) frame**, the bowler is allowed **two more rolls**. Similarly, a bowler scoring a **spare** in the tenth frame is allowed **one more roll**.

   The maximum possible score in a game of bowling (strikes in all ten frames plus two extra strikes for the tenth frame strike) is 300.

   Sample Output:

```
Input first bowl for frame 1: 2
Input second bowl for frame 1: 1
Input first bowl for frame 2: 3
Input second bowl for frame 2: /
Input first bowl for frame 3: 4
Input second bowl for frame 3: /
Input first bowl for frame 4: 2
Input second bowl for frame 4: 1
Input first bowl for frame 5: 7
Input second bowl for frame 5: /
Input first bowl for frame 6: X
Input first bowl for frame 7: 1
Input second bowl for frame 7: /
Input first bowl for frame 8: 7
Input second bowl for frame 8: 1
Input first bowl for frame 9: X
Input first bowl for final frame: 8
Input second bowl for final frame: 1

 Overall Frames:
Frame 1 Frame 2 Frame 3 Frame 4 Frame 5 Frame 6 Frame 7 Frame 8 Frame 9 Frame 10
2 1     3 /     4 /     2 1     7 /     X       1 /     7 1     X       8 1

3       17      29      32      52      72      89      97      116     125
```

4. Books that are published by Penguin House are accompanied with a code which uniquely identifies the book. The code consists of a sequence of 10 decimal digits. However in certain cases, the capital letter X may also appear as the tenth digit. .

   In fact, only the first nine digits in the code are used to recognize a book. The tenth character serves as a check digit to verify that the preceding 9 digits are properly formed. This check digit is selected so that the value computed as shown in the following algorithm is **evenly divisible by 11**. The role of X as the tenth digit is to guarantee the divisibility by 11 due to the need of a check digit that sometimes has to be as large as 10.

   The algorithm used to check the validity of code is relatively simple. Two totals, t1 and t2, are computed over the digits of the code, with t2 being the sum of the partial sums in t1 after each digit of the code is added to it. The code is **correct** if the final value of t2 is evenly divisible by 11.

   An example will visualize how your program should work. Consider the (correct) code: 0-13-162959-X. First look at the calculation of t1:

digits in the code	0	1	3	1	6	2	9	5	9	10(X)
partial sums	0	1	4	5	11	13	22	27	36	46

The calculation of t2 is done by computing the total of the partial sums in the calculation of t1:

t2 (running totals)	0	1	5	10	21	34	56	83	119	165

```
This_is_fun
This_is_fun is incorrect.
0-89237-010-6
0-89237-010-6 is correct.
0-345-31386-0
0-345-31386-0 is correct.
01315-2447-X
01315-2447-X is correct.
0-345-24865-1-150
0-345-24865-1-150 is incorrect.
```

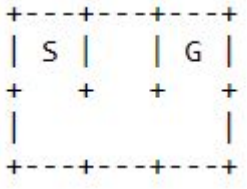
5. Finding a route through a maze is a popular problem for computers. In this problem, a maze will consist of a rectangular array of square cells, each of which may have walls on the north, south, east and/or west sides of the cell. One cell will be identified as the starting point, and another will be identified as the ending point/goall. Your task is to find the **unique path** from the starting point to the goal, label each cell in the path with its sequence in the path, identify the cells that were visited but that were not in the path, and display the maze.

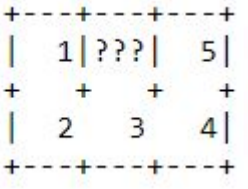
   The algorithm you use to find a path through the maze must be the one described below. Imagine a robot is positioned in the starting cell. The robot first tries to go west from that cell, then north, then east, then south, in sequence. The robot can move in the selected direction if

a. there is **no wal**l preventing it from moving in that direction, and
b. it has not yet been in the next cell in that direction.

   When the robot reaches the ending point, its trip is over. If the robot reaches a cell at which no further progress is possible, it retreats to the previous cell it occupied and attempts to move in the next untried direction.

   Consider the simple maze shown on the left below. It is two cells high and three cells wide. The starting cell is labeled 'S' and the goal cell is labeled 'G'. When the robot starts, it would first try to move west (left), but finds a wall. It then tries to move north (up), and is again blocked by a wall. A wall also prevents it from moving east (right), so it finally tries to move south (down), and succeeds. From the new cell it will eventually move east. Here it repeats its movement algorithm. Although no wall blocks its potential westward movement, it has already 'visited' the cell in that direction, so it next tries to move north, and is successful. Unfortunately, after moving north, it finds no way to extend its path, and so it retreats to the previously occupied cell. Now it tries to move east, and is successful. From that cell it will move north, and there it finds the goal. The maze that would be displayed on the output is shown on the right below. Note that the starting cell is labeled '1', each cell in the path to the goal (including the one containing the goal) is labeled with its sequence number, and each cell that was visited but is not in the path is labeled with question marks.





Input

View the maze as an array of cells, with the northernmost row being row 1, and the westernmost column being column 1. In the maze above, the starting cell is row 1, column 1, and the goal cell is row 1, column 3.

There will be one or more mazes to process in the input. For each maze there will first appear six integers. The first two give the height (number of rows) and width (numer of columns) of the maze (in cells). The next two give the position (row and column number) of the starting cell, and the last two give the position of the goal. No maze will have more than 12 rows or 12 columns, and there will always be a path from the starting point to the goal.

Following the first six integers there will appear one integer for each cell, in row major order. The value of each integer indicates whether a cell has a wall on its eastern side (1) and whether it has a wall on its southern side (2). For example, a cell with no eastern or southern wall has a value of 0. A cell with only a southern wall has a value of 2. A cell with both an eastern and a southern wall has a value of 3. The cells on the periphery of the maze always have appropriate walls to prevent the robot from leaving the maze; these are not specified in the input data.

The **last maze** in the input data will be followed by **six zeroes.**

Output

For each maze, display the maze as shown in the example above and the expected output below, appropriately labeled and prefixed by the maze number. The mazes are numbered sequentially starting with 1.

Sample Output:

```
2 3 1 1 1 3
Maze 1

1 1 0
0 0 0
+---+---+---+
|  1|???|  5|
+   +   +   +
|  2   3   4|
+---+---+---+


4 3 3 2 4 3
Maze 2

0 3 0
0 2 0
0 3 0
0 1 0
+---+---+---+
|??? ???|???|
+   +---+   +
|  3   4   5|
+   +---+   +
|  2   1|  6|
+   +---+   +
|       |  7|
+---+---+---+


0 0 0 0 0 0
```