Homework #3
CS300/500

Objective: Focus on problem solving.

CS300 students choose two of the following problems to solve. **CS500** students choose three of the following problems to solve. Each level can complete one additional problem as a bonus to apply to a lab.

Please include the solution to each problem in a function based on the question number below (a function header has been provided for you for each problem). All of your code should be in one file (i.e., library) so that unit tests can run against the functions during grading. No need to submit a main.cpp.

Please don't forget to include the Independent Completion Form with your assignment.

HW3_01: *Common Permutations*
Given two strings *a* and *b*, print the longest string *x* of letters such that there is a permutation of *x* that is a subsequence of *a* and there is a permutation of *x* that is a subsequence of *b*.

Input
The input file contains several cases, each case consisting of two consecutive lines. This means that lines 1 and 2 are a test case, lines 3 and 4 are another test case, and so on. Each line contains one string of lowercase characters, with the first line of a pair denoting *a* and the second denoting *b*. Each string consists of at most 1,000 characters.

Output
For each set of input, output a line containing *x*. If several *x* satisfy the criteria above, choose the first one in alphabetical order.

Sample Input
```
pretty
women
walking
down
the
street
```

Sample Output
```
e
nw
et
```

HW3_02: *Vito's Family*
The famous gangster Vito Deadstone is moving to New York. He has a very big family there, all of them living on Lamafia Avenue. Since he will visit all his relatives very often, he wants to find a house close to them.

Indeed, Vito wants to minimize the total distance to all of his relatives and has blackmailed you to write a program that solves his problem.

Input
The input consists of several test cases. The first line contains the number of test cases. For each test case you will be given the integer number of relatives r ($0 < r < 500$) and the street numbers (also integers) $s_1, s_2, s_3, \dots., s_i, \dots, s_r$ where they live ($0 < s_i < 30{,}000$).
Note that several relatives might live at the same street number.

Output
For each test case, your program must write the minimal sum of distances from the optimal Vito's house to each one of his relatives. The distance between two street numbers s_i and s_j is $d_{ij} = | s_i - s_j |$

Sample Input
```
3
2 2 5
3 2 4 6
3 5 9 13
```

Sample Output
```
3
4
8
```

HW3_03: *Stacks of Flapjacks*
Cooking the perfect stack of pancakes on a grill is a tricky business because no matter how hard you try, all pancakes in any stack have different diameters. For neatness's sake, however, you can sort the stack by size such that each pancake is smaller than all the pancakes below it. The size of a pancake is given by its diameter.

Sorting a stack is done by a sequence of pancake "flips." A flip consists of inserting a spatula between two pancakes in a stack and flipping (reversing) all the pancakes on the spatula (reversing the sub-stack). A flip is specified by giving the position of the pancake on the bottom of the sub-stack to be flipped relative to the entire stack. The bottom pancake has position 1, while the top pancake on a stack of *n* pancakes has position *n*.

A stack is specified by giving the diameter of each pancake in the stack in the order in which the pancakes appear. For example, consider the three stacks of pancakes below in which pancake 8 is the top-most pancake of the left stack:

8	7	2
4	6	5
6	4	8
7	8	4
5	5	6
2	2	7

The stack on the left can be transformed to the stack in the middle via *ffip(3)*. The middle stack can be transformed into the right stack via the command *ffip(1)*.

Input
The input consists of a sequence of stacks of pancakes. Each stack will consist of between 1 and 30 pancakes and each pancake will have an integer diameter between 1 and 100. The input is terminated by end-of-file. Each stack is given as a single line of input with the top pancake on a stack appearing first on a line, the bottom pancake appearing last, and all pancakes separated by a space.

Output
For each stack of pancakes, your program should echo the original stack on one line, followed by a sequence of flips that results in sorting the stack of pancakes so that the largest pancake is on the bottom and the smallest on top. The sequence of flips for each stack should be terminated by a 0, indicating no more flips necessary. Once a stack is sorted, no more flips should be made.

Sample Input
```
1 2 3 4 5
5 4 3 2 1
5 1 2 3 4
```

Sample Output

```
1 2 3 4 5
0
5 4 3 2 1
1 0
5 1 2 3 4
1 2 0
```

HW3_04: *Primary Arithmetic*
Children are taught to add multi-digit numbers from right to left, one digit at a time. Many find the “carry” operation, where a 1 is carried from one digit position to the next, to be a significant challenge. Your job is to count the number of carry operations for each of a set of addition problems so that educators may assess their difficulty.

Input
Each line of input contains two unsigned integers less than 10 digits. The last line of input contains “0 0”.

Output
For each line of input except the last, compute the number of carry operations that result from adding the two numbers and print them in the format shown below.

Sample Input
```
123 456
555 555
123 594
0 0
```

Sample Output
```
No carry operation.
3 carry operations.
1 carry operation.
```

HW3_05: *The Archaeologist's Dilemma*

An archaeologist, seeking proof of the presence of extraterrestrials in the Earth's past (not Dr. Jones), has stumbled upon a partially destroyed wall containing strange chains of numbers. The left-hand part of these lines of digits is always intact, but unfortunately the right-hand one is often lost because of erosion of the stone. However, she notices that all of them are powers of 2 is obvious. To reinforce her belief, she selects a list of numbers on which it is apparent that the number of legible digits is strictly smaller than the number of lost ones, and asks you to find the smallest power of 2 (if any) whose first digits coincide with those of the list.
Thus, you must write a program that, given an integer, determines the smallest exponent E (if it exists) such that the first digits of 2E coincide with the integer (remember that more than half of the digits are missing).

Input
Each line contains a positive integer N not bigger than 2,147,483,648.

Output
For every one of these integers, print a line containing the smallest positive integer E such that the first digits of 2E are precisely the digits of N, or, if there isn't one, the sentence "no power of 2".

Sample Input
1
2
10
Sample Output
4
8
10

Further Explanation
In the sample above, 4 is the output for 1 because it's the smallest number, E, where 2^E starts with a 1. So, starting at 1, 2^1=2, 2^2=4, 2^3=8, 2^4=16, 2^5=32, 2^6=64, 2^7=128, 2^8=256. 16 is the first number that starts with 1, and E=4 in that problem.

For the second input, 2, the answer is 8 because 2^8 = 256 and nothing smaller than 8 results in a number that starts with 2 with the exception of 2^1, but we're assuming numbers are missing.