CSCE 4123: Programming Challenges

String Examples

UNIVERSITY OF ARKANSAS

http://csce.uark.edu +1 (479) 575-6043 yrpeng@uark.edu

Automated Judge Script

UVA 10188

Human programming contest judges are known to be very picky. To eliminate the need for them, write an automated judge script to judge submitted solution runs.

Your program should take a file containing the correct output as well as the output of submitted program and answer either Accepted, Presentation Error, or Wrong Answer, defined as follows:

Accepted: You are to report “Accepted” if the team’s output matches the standard solution exactly. All characters must match and must occur in the same order.

Presentation Error: Give a “Presentation Error” if all numeric characters match in the same order, but there is at least one non-matching non-numeric character. For example, “15 0” and “150” would receive “Presentation Error”, whereas “15 0” and “1 0” would receive “Wrong Answer,” described below.

Wrong Answer: If the team output cannot be classified as above, then you have no alternative but to score the program a ‘Wrong Answer’.

Automated Judge Script

Input

The input will consist of an arbitrary number of input sets. No line will have more than 100 characters.

Each input set begins with a line containing a positive integer n < 100, which describes the number of lines of the correct solution. The next n lines contain the correct solution.

Then comes a positive integer m < 100, alone on its line, which describes the number of lines of the team's submitted output. The next m lines contain this output.

The input is terminated by a value of n = 0, which should not be processed.

Output

For each set, output one of the following:

Run #x: Accepted

Run #x: Presentation Error Run #x: Wrong Answer

where x stands for the number of the input set (starting from 1).

Automated Judge Script

❑ Sample Input

```
2
The answer is: 10
The answer is: 5
2
The answer is: 10
The answer is: 5
2
The answer is: 10
The answer is: 5
2
The answer is: 10
The answer is: 15
1
1 0 1 0
1
1010
1
The judges are mean!
1
The judges are good!
0
```

❑ Sample Output

```
Run #1: Accepted
Run #2: Wrong Answer
Run #3: Presentation Error
Run #4: Presentation Error
```

Sample Solution


```
from sys import stdin, stdout

run = 1
while True:
    num_lines = int(stdin.readline().strip())
    if num_lines == 0:
        break

    out_1 = ""
    out_2 = ""
    for i in range(num_lines):
        out_1 += stdin.readline()

    num_lines = int(stdin.readline().strip())
    for i in range(num_lines):
        out_2 += stdin.readline()
```

Sample Solution

```
    def remove_alpha(t): return "".join([ch for ch in t if
ch.isdigit()])

    if out_1 == out_2:
        res = "Accepted"
    elif remove_alpha(out_1) == remove_alpha(out_2):
        res = "Presentation Error"
    else:
        res = "Wrong Answer"

    stdout.write("Run #{}: {}\n".format(run, res))
    run += 1
```

Common Permutation

Uva 10252

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

A subsequence of a string is a new string that is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters.

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 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.

Common Permutation

❑ Sample Input

```
pretty
women
walking
down
the
street
```

❑ Sample Output

```
e
nw
et
```

Sample Solution

```cpp
#include …
using namespace std;

int main() {
  string a, b;
  while (getline(cin, a), getline(cin, b)) {
    int aCounts[128], bCounts[128];
    for (int i = 'a'; i <= 'z'; ++i) aCounts[i] = bCounts[i] = 0;

    for (int i = 0; i < a.size(); ++i) ++aCounts[a[i]];
    for (int i = 0; i < b.size(); ++i) ++bCounts[b[i]];

    string longest = "";

    for (int i = 'a'; i <= 'z'; ++i)
      while (aCounts[i] > 0 && bCounts[i] > 0) {
        longest += (char)i;
        --aCounts[i];
        --bCounts[i];
      }

    cout << longest << '\n';
  }
```

File Fragmentation

UVA 10132

Your friend, a biochemistry major, tripped while carrying a tray of computer files through the lab. All of the files fell to the ground and broke.

Fortunately, all of the files on the tray were identical, all of them broke into exactly two fragments, and all of the file fragments were found.

Unfortunately, the files didn't all break in the same place, and the fragments were completely mixed up by their fall to the floor.

The original binary fragments have been translated into strings of ASCII 1's and 0's. Your job is to write a program that determines the bit pattern the files contained.

File Fragmentation

Input

The input begins with a single positive integer on its own line indicating the number of test cases, followed by a blank line. There will also be a blank line between each two consecutive cases.

Each case will consist of a sequence of “file fragments,” one per line, terminated by the end-of-file marker or a blank line. Each fragment consists of a string of ASCII 1’s and 0’s.

Output

For each test case, the output is a single line of ASCII 1’s and 0’s giving the bit pattern of the original files. If there are 2N fragments in the input, it should be possible to concatenate these fragments together in pairs to make N copies of the output string. If there is no unique solution, any of the possible solutions may be output.

Your friend is certain that there were no more than 144 files on the tray, and that the files were all less than 256 bytes in size.

The output from two consecutive test cases will be separated by a blank line.

File Fragmentation

❑ Sample Input

```
1

011
0111
01110
111
0111
10111
```

❑ Sample Output

```
01110111
```