2008 2nd Quarter Programming Test
Korea Wisenut, Inc.
Saturday April 12th, 9:30am to 9:30pm

Overview

There are three problems in this test. In each problem, please provide well-documented source code files, a makefile, and a readme file. All programs should compile in Linux at the assigned test servers. You have to use the specified programming language for each problem. Every problem has either Java or C++ as its designated programming language. You can use standard C++ libraries or Java language libraries. No other third party code is allowed in this test. During the test, you can use your books, use Internet for research, or any other means to come up with proper solutions. However, you should not discuss or share any code, ideas or specific solutions of any problems in this test with other contestants. Any such activities will entirely disqualify the solutions of the involved.

Evaluation

Each problem in this test has a maximum score you can get. Here is how we grade your solution. First, if your program compiles correctly by simply typing "make" at a prompt in any directory after extracting your solution files, and produces expected output without a bug, you get the maximum score on correctness. **If your Makefile does not work, you get 0 point on that problem.** Please make sure that your Makefile works without a fail!

Secondly, after your program passes the correctness, we will measure the time it takes to execute on a random, fairly large input. Based on the relative speed of your program among the programs that passed the correctness test, you will receive a score from 0 to a pre-defined maximum score. If your program fails on the correctness test, you automatically get 0 point on the speed test.

If your program does not compile or fail on the correctness test, the maximum score you can get out of readme and the source code is **20%** of the total score of the problem. Actual score in this case is largely dependent on readme file and clarity of the source code consistent with the architecture described in the readme file.

The total score of this test is **150** points. In order not to fail on this test, you need to get at least 40 points, **which means you need to get at least one question right.**

Submission

Please use Linux tar command to archive your source code, Makefile, Readme files, and a shell script file for Java solution. No object files or executables should be included in the archive. A Linux "tar xvf your_archive.tar" should be able to extract your archive successfully by creating appropriate problem directories. The archive should have the

If you submit your program (via email) before 7:30pm, there is no delay penalty. Beyond that point, you get a penalty of 1 point for every 10 minutes. **No solution is accepted after 9:30pm. You get 0 point for this test if you do not submit your solution by 9:30pm. Please note that you can submit your solution only once.** Any subsequent submissions will be ignored.

Input and Output

Your solution to each problem should have the following usage:

```
Problem_X <input_file> <output_file>
```

in case of C++ solution, and

```
Problem_X.sh <input_file> <output_file>
```

in case of Java solution. A typical shell file contains one line "java –cp classes Problem_X.java $1 $2" or "java Problem_X.java $1 $2"

Your program, under no circumstances, should output any messages, especially debugging messages, to standard output. This may confuse the grading script for this test and you may incorrectly lose some points because of that.

Problem A (70 points, C++)

A paper "Deterministic Skip Lists" is referenced for this problem. Your task is to understand the algorithm in the paper and implement search, insert, and remove functions. *Both search and insert function codes are shown in the paper, but no remove function code.* Your job is to implement the proposed data structure including remove function and create a program using the data structure to process the input and output file as specified below. You are welcome to do additional optimization beyond this paper.

Here is how we evaluate your solution:

```
score = (is_program_correct) ? 50 : (overall_review) + max(0, (21 – (rank_in_speed)));
```

Here, (is_program_correct) is a predicate to determine whether your program works correctly. The definition of ‘correctness’ here contains the documentation/architecture components as described in the Evaluation section. (rank_in_speed) returns a value from 1 to a maximum of all the correct solutions provided by all the contestants. For example, if your program runs the fastest of all, you get a value of 1. If your program runs 10th of all provided correct solutions, you get a value of 10. If your program does not run correctly, it returns 21. (overall_review) returns a value of up to 20% of the maximum score based on your coding style, architecture, and readme file. The execution time for your program is measured by “time” command in Linux.

Input and Output

Input file contains a series of commands to insert/remove/find some strings. Your program should execute the commands as it sees and produce an output file according to each command.

Insert command, denoted by “INS ‘some string’” inserts the “some string” into the data structure. If there is already a duplicate string already in the data structure, you should print “DUP ‘some string’”, otherwise “INS ‘some string’”. Remove command, denoted by “REM ‘some string’” removes the ‘some string” from the data structure. If there is no such string in the data structure, you should print “NONE ‘some string’”, otherwise “REM ‘some string’”. Find command, denoted by “FIND ‘some string’” finds the “some string” in the data structure. If founds, you should print “FIND ‘some string’”, otherwise “NONE ‘some string’”.

Sample Input

```
INS abc
INS bcd
INS cde
FIND abc
FIND lll
REM lll
REM kkk
FIND lll
FIND abc
INS abc
```

Sample Output

```
INS abc
INS bcd
INS cde
FIND abc
NONE lll
NONE lll
NONE kkk
INS lll
NONE lll
FIND abc
DUP abc
```

Please do not assume any limit on the number of commands in the input file. If your program can not allocate more memory, then just print out an error message that “no more available memory” in the output file; at the time of such memory allocation error, you should have executed enough number of commands in the input file. Before exiting, please ensure that you close both input/output files.

Program Usage

```
Problem_A <input_file> <output_file>
```

Problem B (40 points, C++)

A paper “A Linear Sieve Algorithm for Finding Prime Numbers” is referenced for this problem. It is a simple, elegant algorithm to find prime numbers between 2 and n, where n is an input integer value. Your task is to understand the algorithm in this paper and implement it. There are three approaches presented in this paper for implementing the set of integers between 2 to n. You can pick and choose any approach which you think will help produce the fastest execution time.

The evaluation scheme for this problem is the same as the Problem A except the maximum score:

```
score = (is_program_correct) ? 20 : (overall_review) +  max(0, (21 – (rank_in_speed)));
```

Input

Input file has one line of an integer number n.
n // the input number n to specify the set between 2 to n.

Output

Output file should list all the prime numbers you found in the following format.
m // the number of prime numbers you found
p1 // first prime number
p2 // second prime number
....
pm // mth prime number

Please do not assume any limit on the input number n. The maximum value of n is that of unsigned integer type in C++.

Sample Input

```
9
```

Sample Output

```
4
2
3
5
7
```

Program Usage

Problem_B <input_file> <output_file>

Problem C (40 points, Java)

A subsequence of a given sequence S consists of S with zero or more elements deleted. Formally, a sequence $Z = z_1z_2...z_k$ is a subsequence of $X = x_1x_2...x_m$ if there exists a strictly increasing sequence $< i_1, i_2,..., i_k >$ of indices of X such that for all $j = 1, 2,...,k$, we have $x_{ij} = z_j$. For example, Z = bcdb is a subsequence of X = abcbdab corresponding index sequence <2, 3, 5, 7>. Your job is to write a program that counts the number of occurrences of Z in X as a subsequence such that each has a distinct index sequence.

The evaluation scheme for this problem is the same as the Problem A except the maximum score:

score = (is_program_correct) ? 20 : (overall_review) + max(0, (21 – (rank_in_speed)));

Input

The first line of the input contains an integer N indicating the number of test case to follow. The first line of each test case contains a string X, composed entirely of lowercase alphabetic characters and having length no greater than 1,000. The second line contains another string Z having length no greater than 100 and also composed of only lowercase alphabetic characters. Be assured that neither Z nor any prefix or suffix of Z will have more than 2^{32} distinct occurrences in X as a subsequence.

Output

For each test case, output the number of distinct occurrences of Z in X as a subsequence. Output for each input set must be on a separate line.

Sample Input

```
2
babgbag
bag
rabbbit
rabbit
```

Sample Output

```
5
3
```

Program Usage

```
Problem_C.sh <input_file> <output_file>
```