University of Scranton
ACM Student Chapter / Computing Sciences Department
9th Annual High School Programming Contest (1998)

Problem 1: Decimal Representation of Rational Numbers

A rational number is one that can be expressed as a fraction $\frac{k}{m}$, where k and m are integers and m is nonzero. It turns out that these are also the numbers than can be expressed in *repeating decimal* form. That is, every rational number can also be written in the form

$$p.d_1d_2d_3\cdots\overline{d_{n-r+1}d_{n-r+2}\cdots d_n}$$

where p is an integer, each d_i is a digit between 0 and 9, and the bar over the last r digits indicates that that sequence of digits repeats forever.

You are to develop a program that takes as input two nonnegative integers k and m (a numerator and denominator) and that produces as output the *simplest* repeating decimal representation of the rational number $\frac{k}{m}$. By simplest we mean that the number of digits appearing after the decimal point is minimum. For example, the rational number $\frac{13}{6}$ is represented by each of $2.1\overline{6}$, $2.16\overline{6}$, and $2.1\overline{66}$, but only the first of these is in simplest form. As another example, the simplest repeating decimal representation of $\frac{40}{4}$ is $10.\overline{0}$

Your program should terminate when the user enters zero for the denominator.

In order to simplify the task of producing output, your results are to be written in the form $p.d_1d_2\cdots d_n \quad r$, where r indicates the length of the repeating sequence of digits. (See example below.)

You may assume that any rational number used in testing your program is such that its simplest repeating decimal representation has no more than thirty digits appearing after the decimal point.

Example:

```
Enter numerator:  14
Enter denominator:  5
2.80 1

Enter numerator:  131
Enter denominator:  350
0.37428571 6

Enter numerator:  56
Enter denominator:  0
```

University of Scranton
ACM Student Chapter / Computing Sciences Department
9th Annual High School Programming Contest (1998)

Problem 2: Any Way Home?

Imagine a country in which each road provides a one-way path from one city to another. In such a country there may be cities such that, once you have left them, there is no way back!

Write a program that, given as input a table describing the map of the country, identifies those cities to which it is not possible to return once you have left. More precisely, the program should identify any city having the property that there is no path of roads by which you can leave it and then return to it.

The first input is an integer n describing the number of cities in the country. (Cities are numbered from 1 to n.) On the remaining n lines of input is an $n \times n$ table of 0's and 1's describing the map of the country. A one in row i and column j indicates that there is a one-way road from city i to city j. A zero indicates the absence of such a road. (Rows are numbered from top to bottom, columns from left to right, as they appear on the screen.)

You may assume that $n \leq 20$. Repeat until the user enters zero for the number of cities.

Example:

```
Enter number of cities:  8
0 1 1 0 0 0 0 0
0 0 1 0 0 0 0 1
1 0 0 0 0 0 0 0
0 0 0 0 1 0 1 1
0 0 0 0 0 1 0 1
0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 1
0 0 0 0 0 0 0 0

4 7 8

Enter number of cities:  0
```

University of Scranton
ACM Student Chapter / Computing Sciences Department
9th Annual High School Programming Contest (1998)

Problem 3: Shortest Substring Containing a Subsequence

Let S and T be strings of letters. We say that S is a *subsequence* of T if the letters that make up S appear in the same order within T, but not necessarily adjacent to one another. For example, **MOO** is a subsequence of **NONMONOTONIC**, because the letters **M**, **O**, and **O** appear within **NONMONOTONIC** as the fourth, fifth, and seventh letters, respectively. However, **POT** is neither a subsequence of **LIFTS** nor of **ATROPHY**. In the first case, not all of the letters in **POT** appear in **LIFTS**; in the second case, although all the letters of **POT** are present in **ATROPHY**, they are not in the proper order.

You are to develop a program that takes as input two strings S and T of lower case letters and reports whether or not S is a subsequence of T. If it is not, your program should simply print 0; if it is, your program should print the starting and ending positions of the *shortest* substring of T containing S as a subsequence. (Positions are numbered from left to right starting at 1.) If there are two or more shortest substrings of T containing S as a subsequence, identify the leftmost such substring.

Your program should repeat until two empty strings are issued as input. You may assume that no string has length exceeding 80.

Example:

```
Enter S: ben
Enter T: aboekneno
2 6

Enter S: pelth
Enter T: dgiephwrtlx
0

Enter S: sss
Enter T: tstststsstst
6 9

Enter S:
Enter T:
```

University of Scranton
ACM Student Chapter / Computing Sciences Department
9th Annual High School Programming Contest (1998)

Problem 4: String Editor

You are to write a program that accepts as input a single string and then allows the user to enter commands that modify the string. Your program should implement five types of commands: DELETE, REPLACE, INSERT, RESTART, and QUIT.

If the user enters the command DELETE, your program should prompt the user for the starting and ending positions of the substring to be deleted. (Positions are numbered from left to right starting at 1.)

If the user enters the command REPLACE, your program should prompt the user for the starting and ending positions of the substring to be replaced, and then prompt the user for the replacement string. (Note that the replacement string need not be the same length as the one being replaced.)

If the user enters the command INSERT, your program should prompt the user for the position at which to insert a string, and then prompt for the string to be inserted.

For each of the above commands, after the user has responded to the prompt, your program should carry out the command, print the updated string, and then prompt the user to enter another command.

If the user enters the command RESTART, your program should discard the current string and prompt the user to enter a new string.

If the user enters the command QUIT, your program should terminate.

You may assume that the length of the string being edited will never exceed 80 characters.

Example:

```
Enter string:  Existence precedes essence.

Command:  DELETE
Starting position:  11
Ending position:  19

Existence essence.

Command:  INSERT
Starting position:  6
Insertion string:  EE

ExistEEence essence.

Command:  RESTART
```

```
Enter string:  Nowhere
Command:  REPLACE
Starting position:  3
Ending position:  6
Replacement string:  p

Nope

Command:  QUIT
```

University of Scranton
ACM Student Chapter / Computing Sciences Department
9th Annual High School Programming Contest (1998)

Problem 5: Scheduling

You are to write a program that schedules jobs as follows. The program should first prompt for the number of jobs to be scheduled, an integer between 0 and 10. Next the program should prompt for the arrival time and duration of each job, where the arrival time will be an integer between 0 and 19 and the duration will be an integer between 1 and 10. Your program should output a schedule indicating what job should be worked on at each point in time between time 0 and the time when all jobs have finished.

A job cannot be worked on until it arrives, and jobs are prioritized so that shorter jobs will be worked on first. If a job arrives that will take less time to complete than the job currently being worked on, the current job is interrupted and "put on pause" while the shorter job is worked on. Now, it may happen that this shorter job is interrupted by an even shorter job. And so on and so forth. When a job is completed, if others are waiting to be resumed or started, the job needing the shortest amount of time to complete should be worked on next. (Ties are broken by arrival time: the earlier the arrival time, the higher the priority.)

Your output should indicate, as illustrated in the example below, the times at which the jobs are started, interrupted, resumed, and completed. Until the user enters 0 for the number of jobs, your program should repeat.

The complexity of the following example merits explanation: Job 1 arrives at time 3, and since it is the only job available, it is started. When job 2 (with duration 7) arrives at time 4, it takes priority over job 1, because job 1 has 9 remaining units of time. Thus, job 1 is interrupted and job 2 is started. Job 3 arrives at time 7, but since its duration, 5 units, exceeds the time remaining for job 2, 4 units, we do not interrupt job 2. However, job 4's arrival at time 9 causes us to interrupt job 2 (which has 2 remaining units of time, compared to job 4's having duration 1). Job 4 finishes at time 10, at which time we resume job 2, because, among all jobs present (jobs 1, 2, and 3), it has the smallest remaining time to completion. By the time job 2 finishes, job 5 has arrived. Among jobs still present (jobs 1, 3, and 5), job 5 has the smallest remaining time to completion, and thus it is started. When job 5 completes, job 3 is started, because it takes priority over job 1. Finally, job 3 finishes, job 1 is resumed, and it finishes.

Example:

```
Enter number of jobs:  5
Enter arrival time for job 1:  2
Enter duration for job 1:  11
Enter arrival time for job 2:  4
Enter duration for job 2:  7
Enter arrival time for job 3:  7
Enter duration for job 3:  5
Enter arrival time for job 4:  9
Enter duration for job 4:  1
```

```
Enter arrival time for job 5:  11
Enter duration for job 5:  3

Time 2:  start job 1
Time 4:  interrupt job 1, start job 2
Time 9:  interrupt job 2, start job 4
Time 10:  finish job 4, resume job 2
Time 12:  finish job 2, start job 5
Time 15:  finish job 5, start job 3
Time 20:  finish job 3, resume job 1
Time 29:  finish job 1

Enter number of jobs:  0
```

University of Scranton
ACM Student Chapter / Computing Sciences Department
9th Annual High School Programming Contest (1998)

Problem 6: Rectangles and Points

You are to write a program that prompts the user to enter two ordered pairs of real numbers, (x_1, y_1) and (x_2, y_2), each of which represents a point on the cartesian plane. These points are to be interpreted to be opposing corners of a rectangle whose sides are parallel to the coordinate axes. (That is, two sides of the rectangle are horizontal and the other two are vertical).

Your program is then to (repeatedly) prompt the user for an ordered pair of real numbers (x, y) representing another point on the plane. For each such point entered by the user, the program should output the distance from that point to the point on the perimeter of the rectangle which is closest to it.

The third point may be anywhere in the plane, including inside the rectangle, outside the rectangle, or on its perimeter.

The program should terminate when the point entered corresponds to the first corner of the rectangle.

Reminder: The distance between points (u_1, v_1) and (u_2, v_2) is $\sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2}$.

Example:

```
Enter opposing corners of rectangle:
x1:  2.0
y1:  2.0
x2:  5.0
y2:  4.0


Enter a point:
x:  4.5
y:  3.0

Shortest distance is 0.5.

Enter a point:
x:  7.0
y:  7.0

Shortest distance is 3.6055.

Enter a point:
x:  2.0
y:  2.0
```