University of Notre Dame
Association of Computing Machinery
Department of Computer Science and Engineering

1995 Fall Programming Contest

September 24, 1995
1 - 4pm

Notes on the Problem Set

There are 8 problems in this problem set. They are not in any particular order of difficulty — read all the problems before deciding which to do! Not including the title page and this page, there are 12 pages in this handout.

If you have a problem of any kind, please notify a judge immediately.

Be sure to read each problem **very carefully**. The problems should contain enough information to program complete solutions.

Judges will NOT give out helpful hints on any of the programs. Judges will clarify any questions should an ambiguity arise. A Judge may tell you to re-read the problem if there is no ambiguity.

For programming purposes, you may assume that all data will be of the format specified in the problem specifications. That is, no invalid data will be used to judge your program.

NOTE: *No* output should have a final blank line!

Contents

1 Pig-Latin

1.1 Description

You have decided that PGP encyption is not strong enough for your email. You have decided to supplement it by first converting your clear text letter into Pig Latin before encrypting it with PGP.

You are to write a program that will take in an arbitrary number of lines of text and output it in Pig Latin. Each line of text will contain one or more words. A "word" is defined as a consecutive sequence of letters (upper and/or lower case). Words should be converted to Pig Latin according to the following rules (non-words should be output exactly as they appear in the input):

1. Words that begin with a vowel (a, e, i, o, or u, and the capital versions of these) should just have the string "`ay`" (not including the quotes) appended to it. For example, "`apple`" becomes "`appleay`".
2. Words that begin with a consonant (any letter than is not A, a, E, e, I, i, O, o, U or u) should have the first consonant removed and appended to the end of the word, and then appending "`ay`" as well. For example, "`hello`" becomes "`ellohay`".
3. Do not change the case of any letter.

1.2 Sample Input

```
This is the input.
```

1.3 Sample Output

```
hisTay isay hetay inputay.
```

2 Rational Spiral

2.1 Description

Rational numbers can be expressed as the division of two integers. They are also countable. This means that an order can be placed on them just like the natural numbers. But this ordering is not obvious, and is not unique. Here is one way to do it: imagine a plane where the X-axis represents the denominator of a rational number and the Y-axis represents the numerator. Starting at the origin, spiral clockwise away from it. Each integer pair that you encounter will represent a rational number. See the figure below.

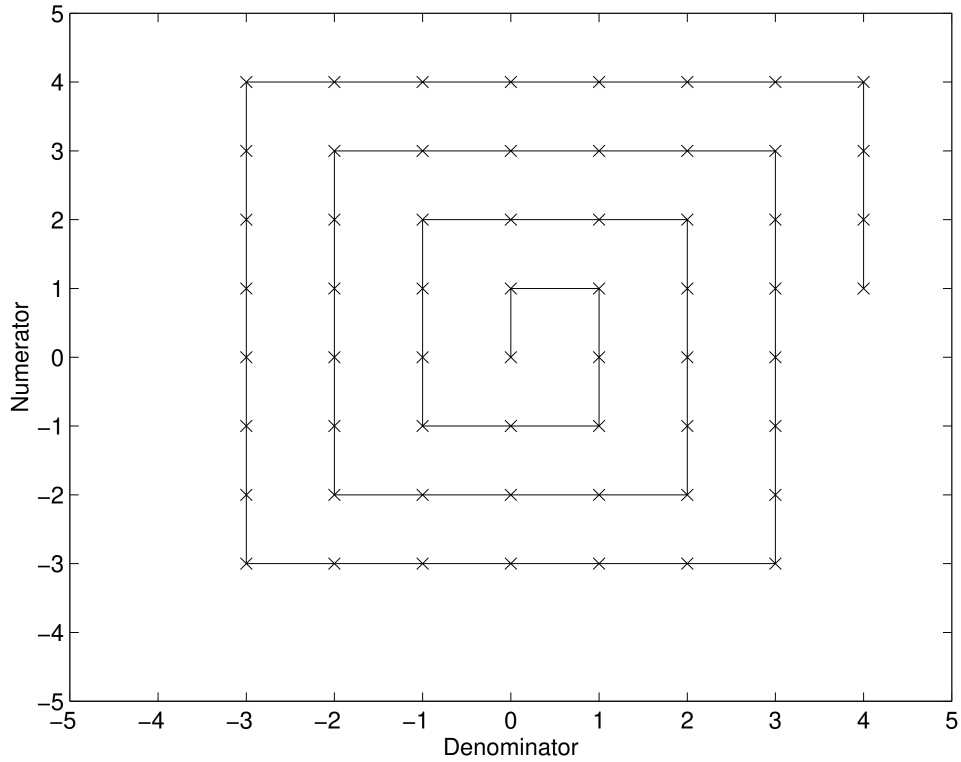

Figure 1: Ordering of the Rational Numbers

The first rational number that you will encounter is `0 / 0`. But wait, this is not a rational number! Therefore, this pair must be skipped. Each time that you pass the Y-axis, the denominator will be `0`, which generates an illegal rational number. You will have to skip all of these.

You will also encounter many repeats. For example $\frac{-1}{1}$ is the same number as $\frac{1}{-1}$. Only count the first occurance of rational numbers in the spiral. Also be aware that numbers can be reduced, which can also cause repeats. For example $\frac{8}{4}$ reduces to $\frac{2}{1}$, so it is a repeat. Be sure to skip all these too.

Given these rules, you can generate an ordered listing of all the rational numbers which does not skip or repeat. Amazing, isn't it?

Your mission, should you choose to accept it, is to order the rational numbers. There will be several lines of input. Each line of input will have a single integer which represents which rational number to output. For example, if a line of input is `4` then print out the `4th` rational number. **Important**, the list starts with the *zeroth* element.

For each line of input, print out the rational number in this form: numerator, followed by a space, followed by a forward slash, followed by a space, followed by the denominator. **Note, if the rational number is**

negative, the minus sign must be printed with the numerator. For example, do not print out $\frac{4}{-1}$, print $\frac{-4}{1}$ even if you are at the (`denominator == -1, numerator == 4`) location on the spiral.

2.2 Sample Input

```
0
1
2
3
10
```

2.3 Sample Output

```
1 / 1
0 / 1
-1 / 1
-2 / 1
3 / 2
```

3 Kindergarten Counting Game

3.1 Description

Everybody sit down in a circle. Ok. Listen to me carefully.

"Woooooo, you scwewy wabbit!"

Now, could someone tell me how many words I just said?

Input to your program will consist of a series of lines, each line containing multiple words (at least one). A "word" is defined as a consecutive sequence of letters (upper and/or lower case).

Your program should output a word count for each line of input. Each word count should be printed on a separate line.

3.2 Sample Input

```
Meep Meep!
I tot I taw a putty tat.
I did! I did! I did taw a putty tat.
Shsssssssssh ... I am hunting wabbits.  Heh Heh Heh Heh ...
```

3.3 Sample Output

```
2
7
10
9
```

4 Fibonacci Freeze

4.1 Description

The Fibonacci numbers (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . .) are defined by the recurrence:

$$
\begin{aligned}
F_0 &= 0 \\
F_1 &= 1 \\
F_i &= F_{i-1} + F_{i-2} \text{ for all } i \geq 2
\end{aligned}
$$

Write a program to calculate the Fibonacci Numbers.

The input to your program would be a sequence of numbers, each on a separate line, specifying which Fibonacci number to calculate.

Your program should output the Fibonacci number for each input value, one per line.

4.2 Sample Input

```
5
7
11
```

4.3 Sample Output

```
The Fibonacci number for 5 is 5
The Fibonacci number for 7 is 13
The Fibonacci number for 11 is 89
```

5 Simply Subsets

5.1 Description

After graduating from the University of Notre Dame, you obtained a job at Top Shelf Software, Inc., as an entry-level computer engineer. On the first day, your manager sits down with you and tasks you with the following job: “We want to see how well you understand computer programming and the abstract science behind it. As an evaluation for all of our new hires, we require them to write a program to determine the relationship between pairs of sets. I’m quite sure that you’ll do well; my confidence is high. Here’s a list of requirements for what the program should do. Good luck.”

Your program should accept an even number of lines of text. Each pair of lines will represent two sets; the first line is represents set A, the second line represents set B. Each line of text (set) will be a list of distinct integers.

After each pair of lines has been read in, the sets should be compared and one of the following responses should be output:

- `A is a proper subset of B`
- `B is a proper subset of A`
- `A equals B`
- `A and B are disjoint`
- `I'm confused!`

5.2 Sample Input

```
55 27
55 27
9 24 1995
9 24
1 2 3
1 2 3 4
1 2 3
4 5 6
1 2
2 3
```

5.3 Sample Output

```
A equals B
B is a proper subset of A
A is a proper subset of B
A and B are disjoint
I'm confused!
```

6 Strategic Defense Initiative

6.1 Description

"Commander! Commander! Please wake up commander!"

". . . mmmph. What time is it?"

"4:07 am, Commander. The following message just arrived on the emergency zeta priority classified scrambler, marked your eyes only."

You grudgingly take the letter, rub the sleep from your eyes, fleetingly wish that the 'Backer closed at an earlier hour, and start to read.

```
``Dear StarWars SDI Commander,

    Bad news, buddy.  Crazy Boris had a bit too much vodka last night
    and when he woke up this morning, instead of the snooze button on
    his alarm clock, he ... well, let me put it this way: we've got
    tons of nuclear missles flying this way.  Unfortunately, all that
    we have is a chart of the altitudes at which the missles are
    flying, arranged by the order of arrivals.  Go for it, buddy.
    Good luck.

                                    Secretary of Defense

    P.S. Hilly and Bill say hi.''
```

To make things worse, you remeber that SDI has a fatal flaw due to the budget cuts. When SDI sends out missles to intercept the targets, every missle has to fly higher than the previous one (go figure). In other words, once you hit a target, the next target can only be among the ones that are flying at higher altitudes than the one you just hit.

For example, if the missles are flying toward you at heights of 1, 6, 2, 3, and 5 (arriving in that order), you can try to intercept the first two, but then you won't be able to get the ones flying at 2, 3, 5 because they are lower than 6. Your job is to hit as many targets as possible. So you have to quickly write a program to find the best sequence of targets that the flawed SDI program is going to destroy.

Russian war tactics are fairly strange; their generals are stickers for mathematical precision. Their missles will always be fired in a sequence such that there will only be *one* solution to the problem posed above.

Input to your program will consist of a sequence of integer altitudes, each on a separate line.

Output from your program should contain the total number of targets you can hit, followed by the altitudes of those targets, one per line, in the order of their arrivals.

6.2 Sample Input

```
1
6
2
3
5
```

6.3 Sample Output

```
Max hits: 4
1
2
3
5
```

7 Polly the Polynomial

7.1 Description

Algebra! Remember algebra? There is a theory that as engineers progresses further and further in their studies, they lose basic math skills. This problem is designed to help you remember those *basic* algebra skills, make the world a better place, etc., etc.

Your program should accept an even number of lines of text. Each pair of lines will represent one problem. The first line will contain a list of integers $\{c_0, c_1, \ldots, c_n\}$ which represent a set of coefficients to a polynomial expression. The order of the polynomial is n. The coefficients should be paired with the terms of the polynomial in the following manner:

$$c_0 x^n + c_1 x^{n-1} + \ldots + c_n x^0$$

The second line of text represents a sequence of values for x, $\{x_0, x_1, \ldots, x_m\}$.

For each pair of lines, your program should evaluate the polynomial for all the values of x (x_0 through x_m) and output the resulting values on a single line.

7.2 Sample Input

```
-2
5 0 1 6
1 -1
7 6 -1
```

7.3 Sample Output

```
-2 -2 -2 -2
6 5 -2
```

8 What's The Frequency, Kenneth?

8.1 Description

```
#include <stdio.h>

main()
{
  int i;
  char *suffix[]= { "st", "nd", "rd" };
  char *item[]= { "Unix" , "cat", "sed", "awk", "grep", "ed", "vi"};

  printf("In the beginning, there was nothing.\n");
  for (i= 0; i < 7; i++)
    printf("And on the %d%s day, God created %s.  And it was good.\n",
           i + 1, (i < 3) ? suffix[i] : "th", item[i]);
}
```

But then God saw that `vi` led people into temptation. Instead of choosing the righteous ways of `make`, `dbx`, and `RCS`, people used long command lines, `printf()`, and tape backups.

So God decreed, "I see that Engineers have thus defiled my `vi`. And so, I shall create emacs, an editor more powerful than words. Further, for each instantiation `vi` hitherto, the Engineer responsible shalt perform Penance. And lo, the Penance wilt be painful; there will be much wailing and gnushing of teeth. The Engineer will read many lines of text. For each line of text, the Engineer must tell me which letters occur the most frequently."

"I charge you all with My Golden Rule: 'Friends shalt not let friends use `vi`'."

8.2 Sample Input

```
When riding your bicycle backwards down a one-way street, if the
wheel falls of a canoe, how many ball bearings does it take to fill
up a water buffalo?
Hello Howard.
```

8.3 Sample Output

Each line of output should contain a list of letters that all occured with the highest frequency followed by the frequency. The list of letters should be an alphabetical list of upper case letters followed by an alphabetical list of lower case letters.

```
e 6
al 7
a 3
Hlo 2
```

Send a message to Columbus:
55 - 27