Third Annual Quinnipiac University Programming Competition for High School Students

Quinnipiac University

April 28, 2007

Instructions

1. Submit a program by attaching it to an e-mail to the address

   `cscjudge@quinnipiac.edu`

   Include the number of the problem you are solving in the subject line of your e-mail.

   Questions may also be addressed to `cscjudge@quinnipiac.edu`; put the word "Question" in the subject line.

2. Attach a single file for your submission, but that file may be a zip file which includes multiple files for your program.

3. You may assume the input is formatted as shown in the problem, and your program must work on such input. You may assume there are no tabs in the input.

4. Your feedback on a submitted problem will be e-mailed in a reply and will be limited to one of the following:

   (a) COMPILE ERROR: The program did not compile

   (b) PROGRAM NOT RUNNABLE: The program compiled, but did not run as a stand-alone program.

   (c) CRASH: The program crashed or threw an exception when it was run

   (d) INCORRECT OUTPUT

   (e) TIMED OUT: The program did not finish in a reasonable amount of time

   (f) ACCEPTED

   If your program crashes, produces incorrect output, or times out, no information will be given as to the input that caused the problem.

1 Football scoring

In football, points may be scored in any of the following ways: a team may get a safety worth two points, a field goal worth three points, or a touchdown worth six points. In addition, after a touchdown, a team either tries for an extra point or for a two point conversion.

Write a program which reads in a string representing the scores for a team and computes the total number of points scored. The characters of the string are as follows:

Character	**Meaning**	**Points**
S	Safety	2
F	Field Goal	3
T	Touchdown	6
X	Unsuccessful extra point	0
+	Successful extra point	1
*	Successful two-point conversion	2

Thus, every `T` will be followed by either an `X`, a `+`, or a `*`. The input string will also be followed by a dollar sign `$` to signify the end of the input. For example, running the program three times might look as follows, with input given in boldface:

```
Enter the scoring, followed by a $: ST+T*TXF$
Total score: 26 points

Enter the scoring, followed by a $: T+T+FT+F$
Total score: 27 points

Enter the scoring, followed by a $: $
Total score: 0 points
```

Remember that a team may fail to score. You may assume there are no spaces in the input.

2 Leap years

The Gregorian calendar was decreed to be the official calendar by Pope Gregory XIII in 1582. It has been in effect in Catholic countries since then, but it was not adopted in the British colonies (including those that later became the United States) until September 1752. Prior to its adoption, the Julian calendar was in effect. According to the Julian calendar, a year is a leap year if it is divisible by 4. The Gregorian calendar follows a similar rule, except for years that are also divisible by 100. Years that are divisible by 100 are only leap years if they are also divisible by 400. Thus, years such as 1992, 1996, and 2004 were leap years because they are divisible by 4 but not by 100. The year 1900 was not and 2100 will not be leap years in the Gregorian calendar because they are divisible by 100, but not by 400, whereas the year 2000 was a leap year because it is divisible by 400.

Write a program that asks the user for a year and computes whether that year was or will be a leap year in Great Britain. You can assume that the input is a valid positive integer less than 4000. For example, if you run the program five times and enter the years 1700, 1752, 1900, 2000, and 2007, your output might look as follows, with input represented by boldface text:

```
Enter a year: 1700
1700 is a leap year

Enter a year: 1752
1752 is a leap year

Enter a year: 1900
1900 is not a leap year

Enter a year: 2000
2000 is a leap year

Enter a year: 2007
2007 is not a leap year
```

Note that 1700 *was* a leap year in Great Britain at the time under the *Julian* calendar.

3 Cryptograms

A *cryptogram* is a puzzle in which a message or quotation is concealed by changing every letter to a different letter of the alphabet, as determined by a key. When the same letter appears twice in the original text, it is encoded each time by the same letter in the encoded text. For example, every `A` in the original text could be changed into an `R`. Similarly, the same letter cannot represent two different letters from the original text, i. e. if an `A` from the original text is changed to an `R`, then no other letter in the original text can be represented by an `R`.

Write a program which takes a key as the first line of input and then uses that key to encode each of the subsequent lines of input. The key consists of the twenty-six letters of the alphabet in some order. The first letter in the key is used to encode the `A`'s, the second letter is used to encode the `B`'s, and so on, until the twenty-sixth and last letter of the key, which is used to encode the `Z`'s. For example, if the key is

```
QWERTYUIOPASDFGHJKLZXCVBNM
```

then each `A` of the original text should be encoded by a `Q`, each `B` should be encoded by a `W`, and so on. Your program should preserve the case of each letter of the message; for example, an uppercase `A` would be represented by an uppercase `Q` and a lowercase `a` would be represented by a lowercase `q`. Also, each character that is not a letter, including punctuation and spaces, should be left intact in the coded message. The program ends when the user enters a blank line.

For example, running the program might look as follows, with the input represented by boldface text:

```
Enter the key (26 letters): QWERTYUIOPASDFGHJKLZXCVBNM
Enter text; use a blank line to end program
We the People of the United States, in Order to form a more
Vt zit Htghst gy zit Xfoztr Lzqztl, of Gkrtk zg ygkd q dgkt
perfect Union, establish Justice, insure domestic
htkytez Xfogf, tlzqwsoli Pxlzoet, oflxkt rgdtlzoe
Tranquility, provide for the common defence, promote the
Zkqfjxosozn, hkgcort ygk zit egddgf rtytfet, hkgdgzt zit
general Welfare, and secure the Blessings of Liberty to
utftkqs Vtsyqkt, qfr ltexkt zit Wstllofil gy Sowtkzn zg
ourselves and our Posterity, do ordain and establish this
gxkltsctl qfr gxk Hglztkozn, rg gkrqof qfr tlzqwsoli ziol
Constitution for the United States of America.
Egflzozxzogf ygk zit Xfoztr Lzqztl gy Qdtkoeq.


Goodbye!
```

You may assume that the key consists only of uppercase letters.

4 Knight moves

On a chessboard, the columns (or files) are labeled with the lowercase letters 'a' through 'h' and the rows (or ranks) are labeled with the numbers 1 through 8. Each individual square is labeled by its letter and its number in that order. Thus, the entire board (from white's point of view) is labeled as in figure 1. A knight moves in an 'L' shape: it either moves two squares left or right and one square up or down, or it moves two squares up or down and one square left or right. Of course, it cannot move off the board. For example, a knight on `b5` can move to any of the squares whose labels are circled in the figure 1.

a8	b8	c8	d8	e8	f8	g8	h8
a7	b7	c7	d7	e7	f7	g7	h7
a6	b6	c6	d6	e6	f6	g6	h6
a5	$\boxed{\text{b5}}$	c5	d5	e5	f5	g5	h5
a4	b4	c4	d4	e4	f4	g4	h4
a3	b3	c3	d3	e3	f3	g3	h3
a2	b2	c2	d2	e2	f2	g2	h2
a1	b1	c1	d1	e1	f1	g1	h1

Figure 1: The squares on a chess board

Write a program which takes no input, but whose output consists of 64 lines. Each line consists of the name of a square followed by a colon, followed by all the squares a knight can reach from that square separated by spaces. For example, the line corresponding to the `b5` square is as follows:

```
b5: a3 a7 c3 c7 d4 d6
```

The first eight lines of output should correspond to the squares from `a1` through `a8` in that order; the next eight lines correspond to `b1` through `b8`, and so on up to `h1` through `h8`. On each line, the reachable squares should appear in the same order on the right side of the colon, as in the above example for `b5`.

5 WFFs

Some time in the 1970's the game *WFF and Proof* was popular. OK, so maybe it wasn't all that popular, but it was a game. The game was based on creating *Well-Formed Formulas* or *WFFs* (pronounced either *woofs* or *wiffs*). A WFF is defined as any of the following strings:

(a) A lowercase p, q, r, or s

(b) An uppercase N followed by exactly one WFF

(c) An uppercase A, C, E, or K followed by exactly two WFFs.

For example, the following table includes a column of WFFs and a column of strings that are not WFFs:

WFFs	**Not WFFs**
p	t (not a legal character)
Nq	Q (the definition is case sensitive)
Crs	pq
$ApNq$	Npq
$KApNqCrs$ (K followed by $ApNq$ and Cqr)	Es (E must be followed by *two* WFFs)
$NKApNqCrs$ (N followed by $KApNqCrs$)	$Esqr$ (only two WFFs can follow an E)

Write a program which prompts the user to enter strings (one per line) and tells the user whether each string entered is a WFF or not. You may assume that the input does not contain any spaces at the beginning or end of the line, though there may be spaces in the middle (which you should ignore). The program ends when the user enters a blank line.

For example, running the program might look as follows, with the input represented by boldface text:

```
Enter possible WFF's; enter a blank line to end program
q
q is a WFF
Q
Q is not a WFF
Np
Np is a WFF
Nrs
Nrs is not a WFF
Cqp
Cqp is a WFF
N   K ApNq Crs
N   K ApNq Crs is a WFF


Goodbye!
```

6 Newspaper deliveries

In Quinnipiac City, avenues run East/West and are numbered beginning with First Avenue along the northern edge of the city, and streets run North/South and are numbered beginning with First Street on the western edge of the city. Bob Cat sells the *Quinnipiac Daily* to newsstands, which are located on every corner of the city. Every morning, he leaves his office at First Avenue and First Street, and delivers newspapers to various newsstands before meeting his brother Tom for breakfast at a location which changes each day, but in all cases is no further south than Twentieth Avenue and no further east than Twentieth Street. Also, the restaurant might be on First Avenue or on First Street, but it is never at the corner of First Avenue and First Street.
He leaves the office early enough to choose which stands to visit; other people deliver to the stands he has omitted. His primary goal is to meet his brother as quickly as possible (and therefore, he only walks south and east) and his secondary goal is to maximize his commission by delivering as many papers as possible.
Write a program which reads in the location of the restaurant where he is meeting his brother and the number of papers ordered by each newsstand between his office and the restaurant and determines the route he uses to make his deliveries.
In the input, the location of the restaurant is given by two numbers on the same line representing the avenue and the street corner, and the number of copies ordered by each stand is then given in a rectangular grid of nonnegative integers. Your output should be a sequence of `S`'s and `E`'s without spaces that represent the route he should take, followed by the total number of newspapers he has delivered. If there is more than one optimal route, any route printed will be accepted.
For example, running the program might look as follows, with the input represented by boldface text:

```
Enter location (avenue and street) of the restaurant: 5 7
Enter the number of copies ordered at each location
3 8 2 1 9 7 6
4 0 1 3 2 5 3
8 6 5 7 1 3 2
9 5 2 3 1 7 8
4 1 3 7 6 5 1
Best route: SSEEESSEEE; 55 papers sold.
```

The output tells Bob that the best way to get to the intersection of Fifth Avenue and Seventh Street is to follow the boldfaced route below.

3 8 2 1 9 7 6
4 0 1 3 2 5 3
8 **6** **5** **7** 1 3 2
9 5 2 **3** 1 7 8
4 1 3 **7** **6** **5** **1**

which would allow him to sell 55 papers.