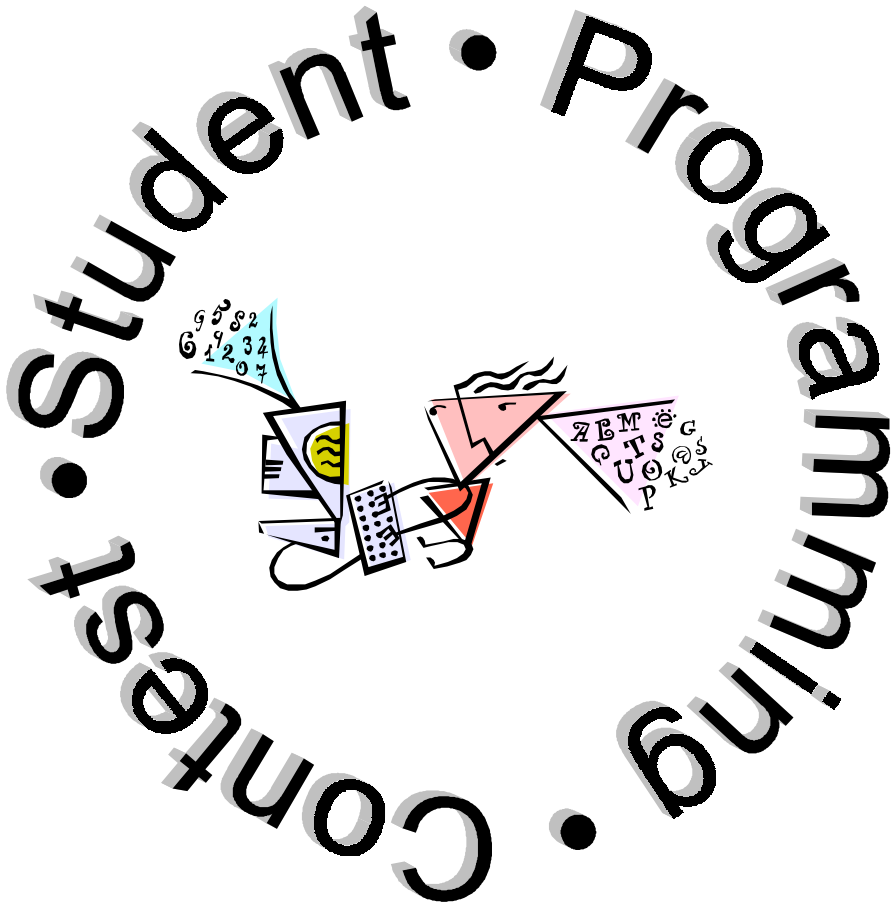
The Seventh Annual

Student Programming Contest

of the

CCSC Southeastern Region



Saturday, November 4, 2000

8:00 A.M. – 12:00 P.M.

The Trouble With Tribbles

The Problem

Kirk: "Get us out of here, Mr. Scott!"
Scotty: "She canna take much more of this, Cap'n!"
Spock: "Interesting."
Kirk: "Report, Bones?"
McCoy: "*#%@ it, Jim, I'm just a doctor!"
Scotty: "She's gonna blow, Cap'n!"
Spock: "Fascinating."
Kirk: *(Grasping the helm with his trademark wild-eyed look)* "Then let it blow. At least we'll take those tribbles with us! At least we'll die free: Ah, freedom, that noble state in which all men yearn to live; that blessed…"
McCoy: *(Grabbing Kirk's arm with his trademark ornery old man look)* "Shut up, Jim!"
(Turning to Spock) "Why don't you do something, Mr. Smarty-Pants-Pointy-Eared-Halfbreed??"
Spock: "It would be illogical to do nothing. In fact, doctor, while you and the others have been shamelessly displaying your human emotions, I have been calculating the amount of time we have remaining until the warp core collapses upon itself. My calculations show that we can stop this catastrophe from occurring if we direct a high-power quantum phase hole in time neutrino subspace blast toward the warp engines in when the tribble count reaches exactly …

Suddenly Spock's eyes roll back in his head and his body goes rigid.

McCoy: "*#%@ it, Spock! Didn't I tell you to keep a little something on your tummy? Low blood sugar isn't something to screw around with! Now look what's happened!"

With Spock unable to speak and move, the crew looks to you to finish Spock's calculations and save the ship. Here's the situation: A nasty colony of tribbles has set up a breeding ground in the warp core of the Enterprise. The trouble with tribbles is their unbelievably fast rate of reproduction. If the tribbles are not stopped soon, the sheer number of them in the engine will cause the ship to explode. Currently there are 2187 tribbles in the engine. Each tribble autonomously reproduces at the rate of 3 per hour; that is, each single tribble splits into three new tribbles once each hour. Now, for reasons not understood (even by Spock), tribbles are vulnerable to high-power quantum phase hole in time neutrino subspace blasts during their 15^{th} hour of reproduction. The crew doesn't know when the tribbles began reproducing, but they do know that it all started with only one tribble. Write a program that tells Kirk when to give the engine a high-power quantum phase hole in time neutrino subspace blast by giving the count of tribbles that would be present after 15 hours of reproduction and the number of hours remaining until the blast must be given.

Sample Input

There is no external input for this problem.

Sample Output

Your program should direct its output to the screen and should look like: (except for where lines break and the spacing around the numbers X and Y)

```
Anonymous Crew Member: "Captain, I have it! Direct the blast at the engine when the
tribble count reaches exactly X. We have approximately Y hours before you must do
this."
```

Al Gore Rhythm

The Problem

After his brush with the Justice Department over fundraising with Buddhist monks, the Vice President devised a plan to ensure that such activities are carried out in a more discrete manner and are kept less noticeable. Realizing that his bid for the presidency demanded more and more money in the campaign war chest, Mr. Gore devised a "rhythm method" for accepting campaign donations from Buddhist monks and yet avoiding the smear that a campaign finance investigation would put on his candidacy. Gore's theory is that if the donations are spaced no less than 28 days apart, there would be no appearance of impropriety.

To help Mr. Gore keep track of when it's okay to accept money from Buddhist monks, you must write a program to automatically scan the White House email logs for messages from "`veep@whitehouse.gov`" addressed to "`buddha@whitehouse.gov`" (the code name for the Al Gore Rhythm Method). Each such email is a secret entry in Mr. Gore's Buddhist monk fundraising dairy. Your program must send Mr. Gore ("`veep@whitehouse.gov`") a reply to each such email advising him when the next donation may be accepted. You should assume that the email was sent the day that the donation was accepted. In keeping with his keen intellect and his heightened sense of secrecy, Mr. Gore refers to a Buddhist monk as a "BM."

Sample Input

Your program must process the White House electronic mail log, stored in the file `prob2.in` as input. The mail log is an ASCII text file which contains a sequence of one or more valid email messages. Each email message consists of a header followed by the body. A header consists of exactly four lines, each of which begins with a unique keyword. The first line is the sender line and begins with the keyword `From` which is followed by a character string representing the email address of the sender. The second line is the recipient line and begins with the keyword `To` which is followed by a character string representing the email address of the recipient of the message. The third line is the date line and begins with the keyword `Date` which is followed by a character string representing the date on which the email message was received. The fourth line is the subject line and begins with the keyword `Subject` which is followed by an arbitrary character string. A single blank space separates the keyword on each header line from the character string that follows. The body begins on the fifth line of the email message. The body of an email message is simply a sequence of one or more lines of text none of which may be blank. Exactly one blank line separates emails from each other in the file. There will be normal White House email interspersed with the Al Gore Rhythm email, but your program should ignore all email other than those from "`veep`" to "`buddha`."

Sample contents of `prob2.in` could appear as:

```
From bill@whitehouse.gov
To buffy@airhead.net
Date Friday, October 6, 2000
Subject Get together
Hey, honeypie. Ole Big Daddy sure is missin you. I'll be a
lookin for you to come around again this weekend. I've got
a hankerin to light up a cuban.
XXXOOO,
Bill

From veep@whitehouse.gov
To buddha@whitehouse.gov
Date Saturday, November 4, 2000
Subject BM
Dear Buddha, I just had a BM. Please advise.

From reno@justice.gov
To bill@whitehouse.gov
```

```
Date Saturday, November 4, 2000
Subject Cubans
Mr. President:
   Regarding your instructions to "get me some more cubans," were you referring to
cigars or people? If it is the former, I will have a box delivered to your office
this afternoon. If it is the latter, I would respectfully suggest that we do not
want another Elian incident (although my strike team in Miami can be deployed in a
matter of minutes). I will await your further instructions.
Regards,
Janet
```

Sample Output

The output of your program must be directed to the screen and must consist of a reply email for each Al Gore Rhythm email found in the log. Each reply must specify the date on which the next donation may be accepted. Your output must be formatted *exactly* as that below, which is the output corresponding to the sample input above.

```
From buddha@whitehouse.gov
To veep@whitehouse.gov
Date Saturday, November 4, 2000
Subject Re: BM

Thank you for advising me of your BM. You may not have
another BM until Saturday, December 2, 2000.
```

Super Freq

The Problem

A character is known to its homeboys as a super freq if it occurs with frequency greater than 15% in a given passage of text. Write a program that reads an ASCII text file and identifies all the English alphabet (A-Z, a-z) super freqs in each line of that text file. Your program must be case insensitive.

Sample Input

Your program must take its input from the text file `prob3.in`. Possible contents of this file could be:

```
Sally sells sea shells by the sea shore.
How now brown cow.
Hey Sam!
```

Sample Output

Your program must direct its output to the screen and be formatted exactly as shown below. Appropriate output for the sample input above would be:

```
Line 1: S is a super freq.
Line 2: O is a super freq. W is a super freq.
Line 3: There are no super freqs.
```

Monotonic Sentences

The Problem

I'm sure you're familiar with monotonous sentences from your CS lectures, but you may not have heard of *monotonic* sentences. The concept is similar to that of a monotonic function. A function f is *increasing* on some interval of numbers I if for every x, y in I, $x < y$ implies that $f(x) < f(y)$. Likewise, a function is *decreasing* on some interval of numbers I if for every x, y in I, $x < y$ implies that $f(x) > f(y)$. A function is *monotonic* if it is either increasing on I or decreasing on I. A sentence, then could be said to be increasing if for every pair of words x, y in the sentence such that x appears before y in the sentence, x is alphabetically less than y. A sentence is decreasing if for every pair of words x, y in the sentence such that x appears before y in the sentence, x is alphabetically greater than y. A monotonic sentence is one that is either increasing or decreasing.

Write a program that identifies monotonic sentences.

Sample Input

Your program must take its input from the ASCII text file `prob4.in`. The file contains a sequence of one or more sentences, one per line. Blanks spaces and the punctuation marks '.' (period), ',' (comma), and '-' (hyphen) can appear before and after words and must not be considered as part of any word. Sample contents of the input file could appear as:

```
All cows eat grass.
How now brown cow.
Zoos – the greatest entertainment around.
```

Sample Output

Your program must direct its output to the screen and identify all monotonic sentences in the input file. The output which corresponds to the sample input above is:

```
Sentence 1 is monotonic.
Sentence 2 is non-monotonic.
Sentence 3 is monotonic.
```

Too Close For Comfort

The Problem

The U.S. Navy conducts training exercises for its small destroyer class ships that simulate close-quarter engagements that might occur just outside a port in a relatively contained area, such as on of the ports within the Persian Gulf. One of these exercises involves a large number of ships (up to 20) trying to rush for the open sea from various points in a bay in the middle of the night while running in "cloaked" mode (no lights with near silent running). This is an obviously difficult maneuver and the possibility of one or more collisions can be very high.

To reduce the likelihood that two ships will collide during this escape exercise, the Navy is implementing a software system that treats the bay area used in the exercise as a two-dimensional Euclidean plane and uses Global Positioning System (GPS) information to fix the location of each participating ship as an (x,y) coordinate in that plane. You are to help the Navy by writing a program that takes a collection of ship positions and identifies the two ships that are the closest to each other. The Navy can then issue a warning to each vessel to modify its course.

Sample Input

Your program must take its input from the text file `prob5.in`. This file contains an unspecified number of fleet position specifications. Each specification begins with a line containing a single integer value N representing the number of ships currently in the fleet ($1 < N < 21$). The subsequent N lines complete the specification by listing a coordinate location (x, y) of each ship in the fleet, one per line from Ship 1 to Ship N. (Note: You may assume that there will be no more than two ships the same distance from each other.) Sample contents of `prob5.in` might be:

```
7
0.7 2
1 2
2 2
1 1.5
2 1
3 1
4 2
8
0.25 1
0.6 0.8
0.5 0.5
1.1 0.5
1 1
2 1
2 0.7
0.9 0.5
```

Sample Output

Your program must direct its output to the screen and format its output in exactly the following form.

```
Fleet 1: Ship 1 and Ship 2 are the closest pair.
Fleet 2: Ship 4 and Ship 8 are the closest pair.
```

Dirt Bag

The Problem

You have just been selected as a contestant on Fox's newest game show "Who Wants to Be a Dirt Bag?" – easily the most popular primetime show in America. The game works as follows: Each contestant is given an empty bag that will hold a certain weight without breaking (the volume of the bag is never an issue, only its weight capacity). The contestant is then given a scoop and a scale and placed in front of several containers of precious stones (diamonds, emeralds, rubies, etc.). Each container is labeled with its total weight and its total monetary value. The contestant is given 30 seconds to fill their bag with precious stones. If the bag's capacity is exceeded it will break, sending the stones spilling across the floor. The studio audience then starts chanting "Dirt bag, dirt bag, you are a dirt bag!" and the smiling host gives the unfortunate contestant a bag of dirt and sends them home. If the bag's capacity has not been exceeded by the end of 30 seconds, the contestant is awarded the monetary value of the stones they put in the bag. Thus, a contestant's strategy is to maximize their profit without exceeding the weight capacity of the bag.

Write a program that computes the optimal loading of the bag for a given game of "Who Wants to Be a Dirt Bag?". Your program should tell what percentage (0% - 100%) of the contents of each container should be loaded into the bag.

Sample Input

Your program must take its input from the text file `prob6.in`. This file contains an unspecified number of game specifications, one per line. Each line begins with an integer C representing the bag's capacity, followed by an integer N representing the number of containers, followed by a sequence of N integers representing the weight of each container, followed by a sequence of N integers representing the value of each container. Thus, each line as 2N + 2 integer values. Sample contents of `prob6.in` might be:

```
105 3 100 10 10 20 15 15
10 5 2 2 6 5 4 6 3 5 4 6
```

Sample Output

Your program must direct its output to the screen and format its output in exactly the following form. For each game specification in the input file, your program must produce the corresponding optimal bag loading. Appropriate output for the sample input above would be:

```
Game 1
------
Container 1: 85%
Container 2: 100%
Container 3: 100%

Game 2
------
Container 1: 100%
Container 2: 100%
Container 3: 33%
Container 4: 0%
Container 5: 100%
```