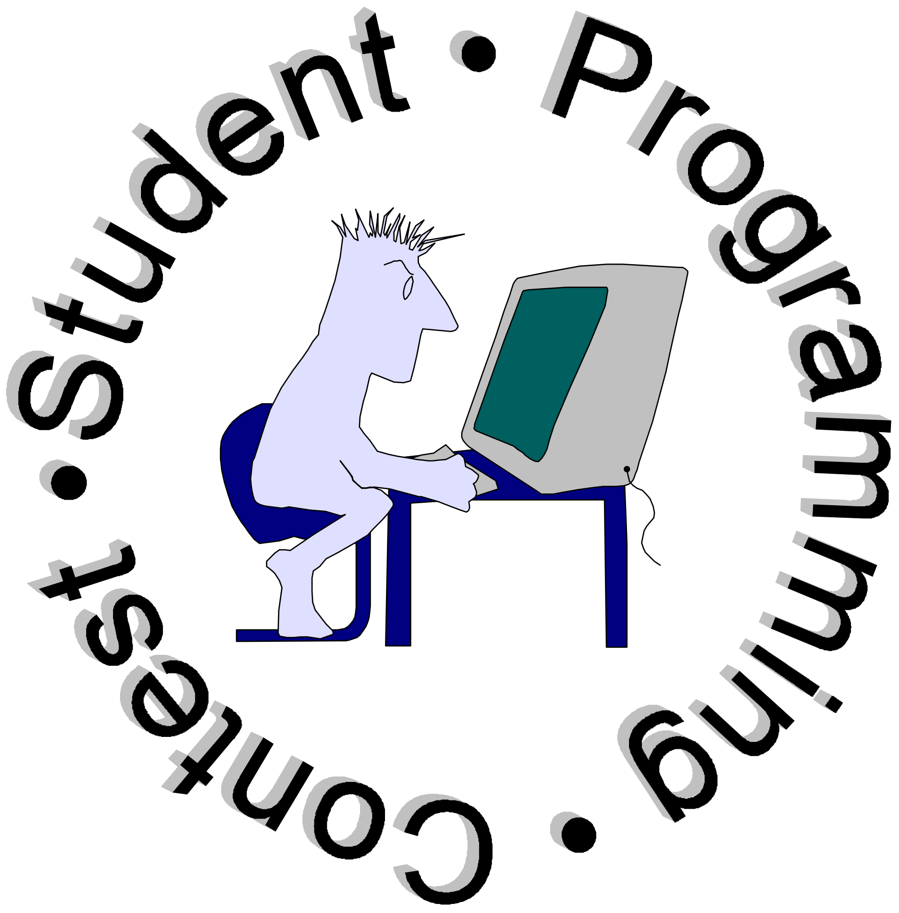
S E S C C C 9 5

The Ninth Annual

Southeastern Small College Computing Conference



Saturday, November 4, 1995
8:00 A.M. – 12:00 P.M.

Swang Business Building
Lipscomb University

N a s h v i l l e

PROBLEM ONE

Crazy Commie Calculator

The Problem

Russian Peasant Multiplication is a technique pioneered by Boris and Igor Stravinsky, two vodka-swilling, Lenin-kissing mathematicians from the former Soviet Union. Being from the old school, Boris and Igor refuse to use modern multiplication methods and insist on using their tried and true approach to multiplying two numbers.

Russian Peasant Multiplication produces the product of two numbers using only multiplication and division by 2. Here's how Boris described it to me last night in the Russian karoke bar near Opryland:

> Ve vill buildski a tableski of tree columnski. Ze finalski entry in ze thirdski columnski is ze desired productski. Initializeski ze firstski columnski to oneski operandski ov ze multiplicationski, ze secondski columnski to ze otherski operandski ov ze multiplicationski. Ze thirdski columnski is usedski toski ...

Well, at this point I stopped listening (it was Gorby night, and a look-alike was up singing Volare). When I got back to my hotel, I worked and worked until I figured out what Boris was saying to me. Here's an example of Russian Peasant Multiplication performing 25 × 20 = 500:

25	20	20
12	40	20
6	80	20
3	160	180
1	320	500

Ze answerski

Sample Input

Your program should take its input from the text file `prob1.in`. This file contains an undetermined number of integer pairs. Your program should produce the product of each integer pair using Russian Peasant Multiplication. Here is a sample of what this file might look like:

```
25 20
20 25
```

Sample Output

Your program should direct its output to the screen. Appropriate output for the sample input listed above would be:

```
25 x 20 = 20 + 160 + 320 = 500
20 x 25 = 100 + 400 = 500
```

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 that text file. Your program should be case insensitive.

Sample Input

Your program should take its input from the text file `prob2.in`. Three examples (A, B and C) of the possible content of this file are shown below:

(A)
```
Sally sells sea shells by the sea shore.
```

(B)
```
How now brown cow.
```

(C)
```
Hey Sam!
```

Sample Output

Your program should direct its output to the screen. Appropriate output for the sample inputs (A, B and C) is shown below:

(A)
```
S is a super freq.
```

(B)
```
O is a super freq.
W is a super freq.
```

(C)
```
There are no super freqs.
```

Bisecting Frogs

The Problem

Professor Harvey Peevey is a research biologist currently on assignment in a dark jungle in South America. Harvey's research involves studying the daily living habits of one of the most reclusive and fascinating animals on earth — the red-eyed nocturnal tree frog (*Visineus Nightus Frogus*).

Long nights of observation have given Harvey insight on many of the frogs' behavior patterns, one of which is their preferred perching height in trees. It seems that a frog's height in a tree on a given night can be expressed as a function of the amount of rain that fell that day in the jungle. The function that Harvey came up with is

$$\mathrm{f}(x) = x^3 + 4x^2 - 10$$

where x is the number of inches of rain for a given day. Evaluating the function *f* at a given *x* yields the height in meters at which the frogs will perch that night.

As you can well imagine, frogs are best observed when perched about midway up the tree — not too high, not too low. Harvey wishes to find out which days are the best for viewing frogs; that is, how much rain will cause the frogs to perch halfway up their tree.

Write a program that finds this optimal amount of rainfall for Harvey and his frogs.

Sample Input

None. However, assume that it rains between 1 and 2 inches each day in the jungle.

Sample Output

Your program should direct its output to the screen. Appropriate output should take the following form:

```
The optimal amount of rainfall for frog viewing is x.xxx inches.
```
(Where `x.xxx` *represents a real number.)*

P R O B L E M   F O U R

Packet Sniffer

The Problem

Sniffed any good packets lately, dude? Hopefully you haven't done this illegally, but rest assured that there are some who have. If you surf the Internet, you have probably visited companies' net sites where they sell their products online: you give them your credit card number and they ship you the goods. That's a convenient way to shop if you can ensure that your credit card number isn't being "sniffed" up by wily hackers and used illicitly.

The Internet is an example of a *packet-switched* network. This means that information is sent in discrete groups of bits called packets from computer to computer. For example, when I send email to someone in the Philippines, my message (at the binary level) is broken up into packets, and routed packet by packet (not all at once) from computer to computer, ultimately to the recipient's computer.

"Packet sniffing" refers to writing a program that grabs the individual packets that come your computer's way and reads their contents. Now the term "packet sniffing" has obvious unethical connotations: usually it refers to writing a program that reads packets addressed to computers other than your own. But the principle is exactly the same when you only intercept those packets that are intended for you.

Let's suppose that a network packet consists of three parts: a special start-of-packet bit pattern, the packet content, and a special end-of-packet bit pattern. Suppose that the start- and end-of-packet pattern are both 1000001 and that the packet content is no more than three consecutive sequences of 7 bits (1's and 0's). So, to write a program to "sniff" packets you need only to write a program that scans its input and "decodes" everything between pairs of 1000001.

For this problem assume that the content of each packet is binary representation of no more than three ASCII characters, each encoded in 7 "bits" (0's and 1's). Your task is to write a program that prints out the message (in English) that is being transmitted by a sequence of packets.

Sample Input

Take your input from the text file `prob4.in`. Here's a sample of its content:

```
10000011100010110111111011111000001100000101000011000001
```

You are guaranteed two things: (1) the encoding is correct, and (2) only lower case letters and punctuation are encoded in the packet.

Sample Output

Your program should direct its output to the screen. The correct output for the sample input above is:

```
boo!
```

Senator Bob

The Problem

As Senator Bob's chief programmer, you have just been faxed a page from the Senator's diary that describes an encryption scheme that Senator Bob wants implemented in software. It seems that Senator Bob needs a method of sending secret messages to a few "professional secretaries" detailing when and where to meet for their next dictation session. Your program should allow both encoding and decoding of the Senator's messages.

Senator Bob's diary page specifies the following method of encoding messages: Take the ASCII value of each character in the message, starting with the last character of the message and ending with the first character of the message, and write this value in reverse order to the encoded message. For example, the character 'A' should be encoded as 56 since it has the ASCII value 65. The character 'z' should be encoded as 221 since it has the ASCII value 122. There should be no spaces separating the numbers in the encoded message. Decoding is just the inverse operation.

Here is a table of the valid ASCII characters appearing in messages:

Char	*ASCII*	*Char*	*ASCII*	*Char*	*ASCII*
A	65	a	97	space	32
B	66	b	98	!	33
...	...	...	...	,	44
...	...	...	...	.	46
...	...	...	...	:	58
Z	90	z	122	;	59

Sample Input

Your program should take its input from the file `prob5.in`. This file consists of an undetermined number of line pairs. The first line in a pair contains either the string "encode" or the string "decode" indicating which operation is to be performed on the associated message. The second line of the pair contains the associated message. You may assume that each message is at most 80 characters long and can contain any upper or lower case letter of the English alphabet plus the space, comma, period, colon, semicolon, question mark, and exclamation mark.

Sample content of the file might be:

```
encode
abc
encode
Have a Nice Day !
decode
998979
decode
```

```
332312179862310199501872379231018117927
```

Sample Output

Your program should direct its output to the screen. This output should contain the result of the indicated operation on each message. Appropriate output for the sample input above would be:

```
Message 1 (encoded): 998979
Message 2 (encoded): 332312179862310199501872379231018117927
Message 3 (decoded): abc
Message 4 (decoded): Have a Nice Day !
```

Palintrees

The Problem

A palindrome is a string containing the same sequence of characters forward as backward. For example, the strings h, madam, xyzyx, and ababa are all palindromes. Let's say that the empty string is not a palindrome.

Define a *palintree* as a binary tree of characters in which at least one root-to-leaf path represents a palindrome. For example, in the tree in Figure 1 there are exactly for root-to-leaf paths but none of them represents a palindrome. Thus this tree is not a palintree. However, in the tree in Figure 2 there is a root-to-leaf path that represents the palindrome ababa; so, this tree is a palintree.

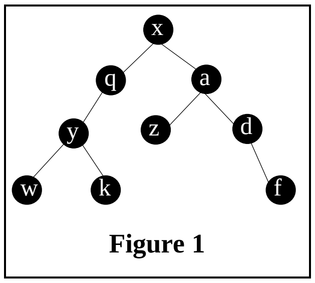


Figure 1

Binary trees are represented in the input with the following form:

tree → empty-tree | (alpha-char tree tree)
empty-tree → ()

Nodes of a tree are specified as single lowercase characters from the English alphabet. There are no blank spaces in the input. So, the palintree in Figure 2 would be represented as

```
(a(b(p(m()())(n()()))())(b(c()())(a(b(a()())())(a()()))))
)
```

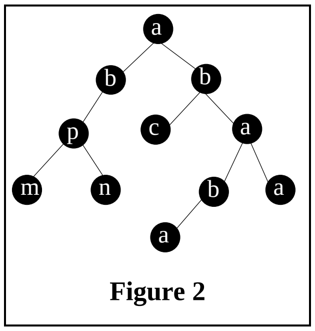


Figure 2

Sample Input

Your program should take its input from the text file `prob6.in`. This file contains an undetermined number of tree specifications, one per line. For example, if the file contained specifications for the trees in Figure1 and Figure 2, its contents would be:

```
(x(q(y(w()())(k()()))())(a(z()())(d(f()())())))
(a(b(p(m()())(n()()))())(b(c()())(a(b(a()())())(a()()))))
```

Sample Output

The output of your program should be directed to the screen. Output from the sample input above should appear as follows:

```
Tree 1 is not a palintree.
Tree 2 is a palintree. Its palindrome is ababa.
```