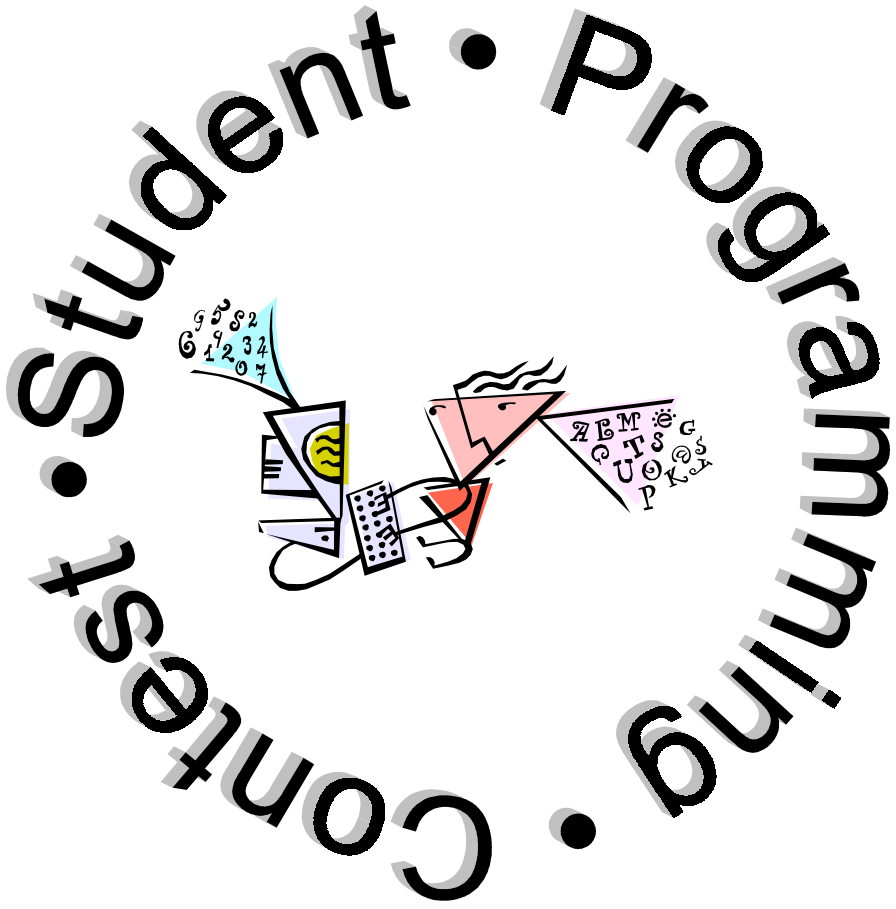
The Sixth Annual

Student Programming Contest

of the

CCSC Southeastern Region



Saturday, November 6, 1999

8:00 A.M. – 12:00 P.M.

Rat Race

The Problem

Write a program that finds a path (if one exists) through a rectangular NxM maze. You must begin in the upper left (0,0) corner and end in the lower right (N,M) corner, while moving only up, down, left, and right (no diagonal moves are allowed). Each maze has at most one path through it.

Sample Input

Your program must take its input from the text file `prob1.in`. This file contains an unspecified number of maze specifications. Each maze specification begins with a pair of integers separated by whitespace on a single line, giving the row x column (NxM) dimensions of the maze. The maze itself is described on the following N lines. Each such line will contain M characters, either X or Y, with X representing a blocked path and Y representing a clear path. Sample contents of `prob1.in` might be:

```
7 5
YYYYY
YXYXY
XXYXX
XYYYY
YYXYX
YXYYY
YYYXY
2 2
YX
XY
```

Sample Output

Your program must direct its output to the screen and must format its output exactly as shown. If a path exists, it must be shown in the output. Specifically, each step in the path must be labeled with a 0 (zero) while each maze location not in the path must be labeled with – (a dash). If there is no path through the maze, "No path exists" should be output. All mazes should be labeled in the output according to their position in the input file (i.e., Maze 1, Maze 2, etc.) Appropriate output for the sample input above would be:

```
Maze 1:
   000--
   --0--
   --0--
   -00--
   00---
   0-000
   000-0

Maze 2:
   No path exists.
```

Piles 'O Coins

The Problem

Suppose you have a bag of N gold coins, each labeled with the integers 1 to N. The coins are labeled according to their weight (and hence their value). That is, a coin labeled '7' weighs 7 units. Write a program that divides the N coins into two piles such that the piles have as near the same value as possible.

Sample Input

Your program must take its input from the text file `prob2.in`. The file contains an unspecified number of lines each containing a positive integer N representing a bag of N coins labeled 1 to N. Sample contents of `prob2.in` might be:

```
15
8
243
```

Sample Output

Your program must direct its output to the screen and format its output in exactly the following form. Appropriate output for the first two lines (in the interest of space on the page) of the sample input above would be: (Notice that the exact composition of the piles isn't necessarily unique. Also: The contents of each pile must be displayed in ascending order of value.)

```
Number of coins: 15
Pile 1 contains 3, 5, 6, 9, 10, 12, 15 with value 60.
Pile 2 contains 1, 2, 4, 7, 8, 11, 13, 14 with value 60.

Number of coins 8:
Pile 1 contains 1, 4, 5, 8 with value 18.
Pile 2 contains 2, 3, 6, 7 with value 18.
```

Palintrees

The Problem

A palindrome is a string containing the same sequence of characters forward as backward. For example, the strings h, madam, xyzyx, aba ababa are all palindromes. Assume that the empty string is not a palindrome. A palintree is a binary tree of characters in which at least one root-to-leaf path represents a palindrome. For example, the tree in Figure 1 contains exactly four root-to-leaf paths but none of them represent palindromes. Thus, the tree in Figure 1 is not a palintree. The tree in Figure 2 does contain a root-to-leaf path that is a palindrome, and thus this tree is a palintree.

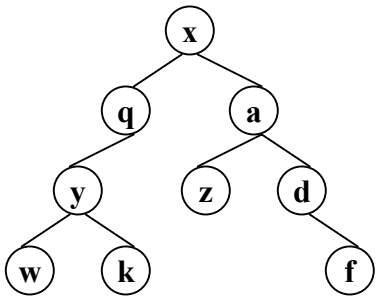


Figure 1

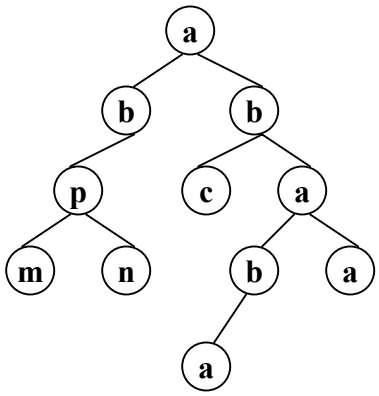


Figure 2

Write a program that decides if a binary tree of characters is a palintree or not.

Sample Input

Your program must take its input from the text file `prob3.in`. This file contains an unspecified number of lines, each of which contains exactly one binary tree specification. Each binary tree specification is of the following form and there are no blank spaces on a line.

tree → empty-tree | (character left-tree right-tree)
empty-tree → ()

Sample contents of `prob3.in` might be (corresponding to Figure 1 and 2 respectively):

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

Sample Output

Your program must direct its output to the screen and format its output in exactly the following form. Appropriate output for the sample input above would be:

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

As I Draw This Magic Square…

The Problem

A magic square is a square matrix in which the sums of the elements in any given row, any given column, and both major diagonals are all equal. Write a program that decides if a square is magic or not.

.

Sample Input

Your program must take its input from the text file `prob4.in`. This file contains an unspecified number of square specifications. Each square specification begins with a single integer value N on a line by itself. This is followed by N lines with N integers each describing the matrix. Sample contents of `prob4.in` might be:

```
2
1 1
1 1
5
3 20 7 24 11
22 14 1 18 10
9 21 13 5 17
16 8 33 12 4
15 2 19 12 22
```

Sample Output

Your program must direct its output to the screen and format its output exactly as shown. Each magic square in the input must generate the string “HOCUS” in the output. Each square that is not magic must generate the string “POCUS” in the output. Appropriate output for the sample input above is:

```
HOCUS
POCUS
```

Boxcar Shunting

The Problem

A freight train has N boxcars, each of which is to be left at a different train station. Assume that the stations are numbered 1 through N and that the freight train visits these stations in the order N through 1.The boxcars are labeled by their destination. To facilitate their removal from the train, the boxcars must be arranged so that they are in the order 1 to N from the front of the train to the back of the train. When the boxcars are in this order, the last car is detached at each station.

The boxcars can be rearranged at a shunting yard that has an input track, and output track, and K holding tracks between the input tracks and output tracks. The N boxcars in the freight train enter the shunting yard on the input track in a random order and leave on the output track in the proper ascending order. Due to the physical nature of the railroad tracks, there are only two moves possible in the shunting yard: (1) A boxcar may be moved from the front (right end, where boxcar 3 is in Figure 1) of the input track to the top of one of the holding tracks or to the rear (left end, where boxcar 9 is in Figure 2) of the output track. (2) A boxcar may be moved from the top of a holding track to the rear of the output track. Figure 1 depicts a train with 9 cars arriving on the input track of a shunting yard with 3 holding tracks. Figure 2 depicts the same train leaving the shunting yard after having its cars rearranged in proper ascending order of destination.

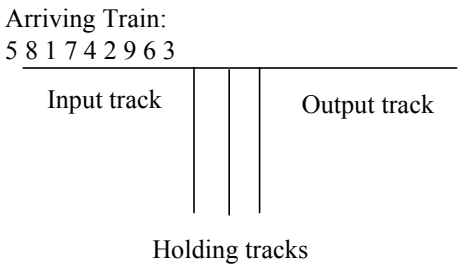


Figure 1

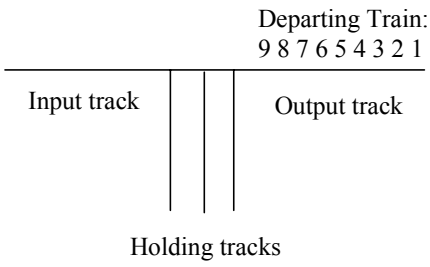


Figure 2

Write a program that decides if a given freight train with N boxcars can be rearranged in a shunting yard with K holding tracks.

Sample Input

Your program must take its input from the text file `prob5.in`. This file contains an unspecified number of boxcar shunting specifications, one per line. Each line begins with an integer N representing the number of boxcars in the train, followed by an integer K representing the number of holding tracks, and ends with a sequence of N integers with values from 1 to N representing the order of the boxcars in the train. Sample contents of `prob5.in` might be:

```
9 3 5 8 1 7 4 2 9 6 3
5 2 1 5 4 3 2
```

Sample Output

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

```
Train 1 can be rearranged.
Train 2 cannot be rearranged. At least 4 holding tracks would be needed.
```

A Ship-Load of Stuff

The Problem

A large ship is to be loaded with cargo that has been packed into N containers. All the containers are the same size, but due to differences in the cargo the containers may have different weights. The ship has a fixed weight capacity that cannot be exceeded. Thus, the dockworkers must ensure that as many containers as possible are loaded without exceeding the ship's capacity.

Write a program that loads a series of ships with cargo in a weight-optimal manner.

Sample Input

Your program must take its input from the text file `prob6.in`. This file contains an unspecified number of ship cargo specifications, one per line. Each line begins with an integer C representing the ship's capacity, followed by an integer N representing the number of containers available for loading, and ends with a sequence of N integers representing the weights of each container Sample contents of `prob6.in` might be:

```
400 8 100 200 50 90 150 50 20 80
10000 3 100000 10000 1000000
```

Sample Output

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

```
Ship 1
------
Number of containers loaded: 6
Weight under capacity: 10

Ship 2
------
Number of containers loaded: 1
Weight under capacity: 0
```