CSE 214 - Data Structures
Fall 1997
Program #3

Due: Monday, October 27, 1997

As we have seen, simulations provide insight into complex problems. Simulation is used in economics, engineering, and the physical sciences because it is often not possible to experiment on the actual entity. For example, an economist cannot play with the US Budget Deficit and see how long it takes the economy to collapse. Rather, he/she will make a computer model and study the effects on it. The significance of simulation results depends on the accuracy of the model as well as how correctly the model has been turned into a computer program.

For your next program, you will write a simulation of the scoring system used in American Jai-alai frontons. Jai-alai is a sport where teams hurl a ball at tremendous speed against a wall, which the other team must catch in a basket or cesta and return it. The first team to miss loses the point. The sport is exciting to watch, but of interest to us because in several states you are allowed to bet on the winners. By simulating the scoring system maybe we can get an 'edge' to aid us in making good bets.

The scoring system is as follows. There are eight teams, numbered 1-8. Initially teams 1 and 2 play, with teams 3-8 in a queue or line. The winner of a point gets to play again, while the loser has to go to the end of the line. The first in the queue is the winner's new opponent. This process repeats until one team gets at least seven points. Since this system favors the low numbered players by giving them more chances to play, the scoring is modified as follows. The first time through the queue, each point is worth a score of 1. After player eight has played his first point, each score is worth 2. Thus player two would have to beat his first seven opponents to win without going back to the queue, where player eight would only have to win his first four. Since it is rare for a player wins that many in a row, the system is supposed to even out. Does it?

Your program is to simulate a specified number of games, with the winner of each point being decided by a random number and thus has a 50-50 chance of going to either player. It will print a list of all trifectas which come up more often than average, where a trifecta is a bet of who will come in first, second, and third in that order. Each time your program simulates a game of jai-alai, find which the three highest players are, and add one to a count of the times this combination has come up. Use a 3-dimensional array to maintain the count. Since there are 8*7*6 different possible trifectas, the average number which should come up are the number of games divided by 336.0.

Although all we are really interested in is the count of which trifectas are most likely to occur, we need to have some reason to believe that our simulation is accurately modeling the game. If our results just look correct, it could be a very expensive trip to the fronton! Thus, we have to be careful to check that the program is playing legal Jai-alai, and we should have a test run where we print out what happens on each point, how much it adds to the score, and who finished where for at least a couple of games and then CHECK VERY CAREFULLY that the results are correct.

To make your task easier, I will make available some subprograms which you are to use in your program. They will be available in ~cse214/program3. The main function, winpoint, takes as arguments two player numbers and uses a random number function to return which one wins the point.

Although the winner is the first person to get to 7 points, there can be ties for second and third. In case of tie, the favorable decision will go to the highest numbered player with the point total, since they got fewer chances to play.

The following sample run should demonstrate the scoring system on a run of just one game with the debug input on. Note that this is a valid game, but that your first game will be different.

```
1 PLAYS          2
1 WINS THE POINT, GIVING HIM          1

1 PLAYS          3
3 WINS THE POINT, GIVING HIM          1

4 PLAYS          3
3 WINS THE POINT, GIVING HIM          2

5 PLAYS          3
3 WINS THE POINT, GIVING HIM          3

6 PLAYS          3
3 WINS THE POINT, GIVING HIM          4

7 PLAYS          3
3 WINS THE POINT, GIVING HIM          5

8 PLAYS          3
8 WINS THE POINT, GIVING HIM          1
```

```
        8 PLAYS         2
        2 WINS THE POINT, GIVING HIM            2

        1 PLAYS         2
        2 WINS THE POINT, GIVING HIM            4

        4 PLAYS         2
        2 WINS THE POINT, GIVING HIM            6

        5 PLAYS         2
        5 WINS THE POINT, GIVING HIM            2

        5 PLAYS         6
        5 WINS THE POINT, GIVING HIM            4

        5 PLAYS         7
        7 WINS THE POINT, GIVING HIM
        3 PLAYS         7
        7 WINS THE POINT, GIVING HIM            4

        8 PLAYS         7
        7 WINS THE POINT, GIVING HIM            6

        1 PLAYS         7
        7 WINS THE POINT, GIVING HIM            8

WIN-PLACE-SHOW IS           7           2           3

BETTER THAN AVERAGE TRIFECTAS:          1 TRIALS

  WIN  PLACE SHOW   OCCURRENCES
    7       2           3                   1
```

You are to turn in the results of two runs of this program. The first plays 5 games and has the debug output on so we can check that your program accurately simulates jai-alai. The second plays 1000 games without the point by point description to get the results we need. It is a waste of computer time to run it on 1000 games before you are sure it works correctly. The number of games to play and whether you want detailed output should be read from standard input.

Challenge Problems

Each assignment will contain one or more extra credit problems for students who want a challenge, or to pursue the subject in greater depth. NOTE THAT THESE PROBLEMS ARE NOT REQUIRED.

(1) This simulation does not give us exact answers of the probability that each distinct trifecta wins (assuming players of equal ability), because we are only getting a random sample of all possible outcomes. However, the search space for all possible jaialai-game outcomes is sufficiently small that you can exhaustively compute them.

Each simulated point yields exactly one of two winners. This defines a ‘‘game tree’’ of all possible outcomes of jaialai games. On this left subtree, we trace all outcomes where player 1 won the point, while the right subtree traces all outcomes where player 2 won the point. Write a recursive program which implicitly builds and traverses this tree, so that you compute the exact probabilities that each trifecta wins. How do the results compare to your randomized simulation?

Rules of the Game

(1) This program is, in my opinion, less difficult than the previous one. But observe that you have less time to complete it. Thus do not wait until the last minute to start work on it.

(2) By starting early, you will have time to attempt the extra credit program. I strongly encourage you to consider this extra credit program, as it is particularly interesting and instructive.

(3) Verifying the correctness of a simulation is difficult. If you do not understand the algorithm completely, your program will never work correctly (I say this each time, but no one believes me!). Thus think about it, work through the above example and be 100% SURE that you can duplicate the scoring algorithm.

(4) Don’t forget to write out and sign the pledge - “On my honor as a student I have neither given nor received aid on this programrq. Your program will not be graded without it.

(5) Before you invest your student loan at the nearest fronton (Milford), remember this simulation is not completely accurate as a different, harder to program, method is used to pick who finished second and third. Also, 1000 trials are not enough to be sure the results are for real. If you are motivated to go beyond the 1000 trials for your own interest, do it at a time the machines are not heavily loaded.