CSC 180 Assignment 2: supplementary information on randomization

Your program will use a ‘‘pseudo-random number generator’’ to create the initial configuration of the deck of cards. A pseudo-random number generator is an algorithm whose returned values are not obviously correlated with anything else and don’t look very algorithmic, even though the numbers returned are not in fact random because they are actually the output of a deterministic algorithm.

The required quality of random number generation and the desired statistical properties of the resulting numbers varies depending upon the domain. For game-playing, the quality requirements are fairly low, although the user should not feel that the numbers are predictable.

In this assignment you will use a random number generator known on unix simply as ‘‘random()’’. It is a fairly good random number generator, and in particular it has the property that all bits of the returned values are random. It returns an integer from 0 to $2^{31} - 1$. You will normally want something more modest, such as an integer from 3 to 10. You can generate an integer from 0 to $i - 1$ inclusive with the expression ‘‘random() % i’’; thus you would use ‘‘random() % 8 + 3’’ to generate an integer from 3 to 10 inclusive (0+3 to 7+3).

This usage really relies on the fact that all bits are random. For example, random()%2 is simply the least-significant bit. Randomness which is only of good quality in the higher bits would not help in this case.

For the card-shuffling algorithm below, where you only need to make a binary (two-valued) decision, you can use ‘‘random() % 2’’. This results in a value of either 0 or 1. This can thus be treated as a boolean value, e.g.

```
if (random() % 2) {
    ...
```

The random() package needs some data to ‘‘seed’’ its random number generator; otherwise it produces the same results each time. It is, like everything else in a computer, algorithmic and deterministic. The seed data is provided by calling a companion function srandom(). In ~ajr/a2/seed.c there is a function ‘‘seed()’’ which you can use to call srandom() with some time-of-day information. This will differ between adjacent invocations of your program so will provide the user with an appearance of randomness. Don’t link with this file; copy it in to your war.c in an appropriate place and edit as desired.

Random() and srandom() are declared in <stdlib.h>. They are in the C library, so you do not need any additional libraries in your cc command. The supplied seed() function also requires the inclusion of <time.h> for its use of the time() function and the data type *time_t*. Time() is also in the C library.

Beware that random() returns type *long*, not *int*.

Shuffling

To ‘‘shuffle’’ an array of *n* numbers, we can traverse all pairs of array cells and randomly either interchange them or not (this is the boolean decision discussed above). So the loop looks something like this:

```
for (i = 0; i < n; i++)
    for (j = i + 1; j < n; j++)
        possibly interchange elements a[i] and a[j]
```

To interchange two elements, we need to assign one of them to a temporary variable, often called ‘‘t’’. Code to interchange the values of *x* and *y* looks something like this:

```
int t;
t = x;
x = y;
y = t;
```

Trace this code to verify that it does in fact set *y* to the previous value of *x*, and also set *x* to the previous value of *y*, for any initial values of *x* and *y*.