Lecture 18

CSE 110

4 August 1992

1 The Tower of Hanoi

As promised, here's the code for the Tower of Hanoi program.

In the program, the three pegs are represented internally by the numbers 1, 2, and 3. As written, the program reads a number of rings from the user, and print out instructions to tell the user how to transfer a tower of this height from ring 1 to ring 3.

The workhorse function, `hanoi`, moves a tower of `num_rings` rings from peg `start_peg` to peg `end_peg`. The first thing it does is to figure out where the spare peg is; to do this it uses a simple trick: If pegs s and e are the start and end pegs, then peg $6 - s - e$ is the spare peg. (For example, if s is 2 and e is 1, the spare peg is $6 - 2 - 1$ or 3.)

If the height of the tower that `hanoi` is called upon to move has size 0, it returns immediately, because it doesn't need to do anything. Otherwise, it calls itself recursively to move the subtower of `num_rings` $- 1$ rings from the start peg to the spare beg, calls `move` to move the largest ring from the start peg to the end peg, and then calls itself recursively again to move the subtower from the spare peg to the end peg.

```
void
  hanoi(int num_rings,
        int start_peg,
        int end_peg)
{
  int spare_peg = 6 - start_peg - end_peg;
```

```
  if (num_rings > 0) {
    hanoi(num_rings - 1, start_peg, spare_peg);
    move(num_rings, start_peg, end_peg);
    hanoi(num_rings - 1, spare_peg, end_peg);
  }

  return;
}
```

`move` is the function which is called each time we want to move a particular single ring. If we were writing our Tower of Hanoi program to do a fancy screen display which showed the rings flying around from peg to peg, we would put the code for drawing the rings on the screen in `move`. In this simple program, we'll be content to just print out an instruction about what peg should be moved where:

```
void move(int ring_num, int start, int end)
{
  printf("Move disk %d from peg %d onto peg %d.\n",
         ring_num, start, end);

  return;
}
```

We'll add a `main` which examines its command-line arguments to decide how many rings the user wants, and then just calls `hanoi` to print the instructions for moving a tower of that many rings from peg 1 to peg 3:

```
void hanoi(int num_rings, int start_peg, int end_peg);
void move(int ring_num, int start, int end);


int main(int argc, char **argv)
{
  int num_rings;

  if (argc != 2) {
    fprintf(stderr, "Usage: %s number_of_rings\n", argv[0]);
    return 1;
  }

  num_rings = atoi(argv[1]);

  hanoi(num_rings, 1, 3);
```

```
  return 0;
}
```

The `atoi` function is something new: It accepts a string which is supposed to contain the string representation of an integer, and it returns the integer that the string represents. For example `atoi("119")` returns the integer 119. `atoi` returns 0 if there is an error, for example, because the string passed in was not composed of digits.

The program is almost trivial with recursion, but it would be very difficult to do without recursion, because the solution we have, to reduce the problem to a simpler case, is already organized along recursive lines.