# CS 104 (Fall 2013) — Assignment 2
# Due: 09/17/2013, 11:59am

BitBucket directory name for this homework (case sensitive): HW2

**(1)** For this problem set, you should start using the "make" utility that is part of Unix/Cygwin. In order to compile all files correctly etc., create a makefile so that your TAs only have to type "make all" at the command line and everything will compile correctly for them. (Besides simplifying our lives, this is the way you should be compiling larger projects in the future anyway.)

**(2)** Review Chapters 2, 4, 5 from the textbook.

**(3)** Imagine that you are an ant crawling around on a 20-sided die.[1] Each of the sides has one number between 1 and 20. The ant starts on the side labeled 1. It wants to visit each of the sides of the die exactly once. It can only move from a side to an adjacent side. When moving from a side labeled $x$ to one labeled $y$, it loses an amount of energy equal to $|x - y|$. (That's the absolute value.) So the ant prefers not to jump up and down in values a lot, but rather visit similarly labeled sides close to each other, if possible. Among all possible ways of visiting all sides, you should find the way using minimum total energy (summed up over all transitions). Write a program that solves thie problem. It doesn't need to output the actual order in which to visit the sides, but it needs to compute the total energy the ant consumes in an optimal visit order.

You almost certainly want to use recursion with backtracking for this. It may be possible to do this otherwise, but it would most likely be much harder.

For full credit, you only need to solve it for the standard 20-sided die (which you can hardcode or describe in a file that you read in). But if it amuses you, you are welcome to extend your solution to allow arbitrary assignments of numbers to the faces of the die, which can be read from a file.

**(4)** For the previous problem, if you are trying all different orders of visiting the faces, starting from 1, give an upper bound on how many different orders you would be trying, by doing some simple calculations. Give a justification for your calculations.

Then instrument your code (by keeping track in a global counter) to figure out how many orders your program *actually* tried, and report that number. This number should be lower than what you calculated as an upper bound. See how close or far apart those numbers are.

**(5)** Implement (by yourself, without using standard template libraries or Vectors or other tools) an implementation of linked lists of strings. Your list should start out empty and have a function for adding a new string at the end of the list.

**(6)** Use your solution to the previous problem to do the following: Read a list of strings (one per line) from a file, and output them in the opposite order. Your file should *not* contain a first line telling you how many strings there are, but just have one string per line. The output should then first print the string from the last line, then the second-last, etc., until you finally output the string on the first line.

You should solve this by outputting the elements of a singly linked list in reverse order. For doing that, write a recursive function. (This is by far the easiest way to do it.)

Note that this problem — outputting the elements of a singly linked list in reverse order — is one of the most commonly asked interview questions. This means that you almost certainly want to have it down pat. It also means that you can likely find solutions online. Please don't cheat — you'd only deprive yourself of having figured this out and knowing it well for when you actually need it.

[1] For those of you who have no experience with 20-sided dice, e.g., from playing D&D or other games, Google what they look like, as you'll need the precise layout.