# Challenge Problems in Algorithms and Data Structures

The purpose of these problems is to provide a challenge to the top students, meaning those who master the regular material of the course and would like to learn more. The problems are due Saturday April 27.

Do not try to work these problem to improve a poor grade. If you are doing badly in the course, concentrate on the regular material.

## 13 Coin Balance Problem.

The 13-coin problem is to use a balance scale three times to find a counterfeit coin among 13 coins, and determine whether it is too heavy or too light. Each step consists of placing a set of coins on each of the two trays, and observing one of three outcomes: the left set is heavier, the right set is heavier, or they weigh the same. The Wikipedia page on Balance Puzzles states that there is no algorithm for this problem, but gives no proof. Prove it.

## Sort a Linked List.

Sort a linked list in $O(n \log n)$ time with $O(\log n)$ workspace. For this problem, you need to write a C++ program and send it to me in a format that I can run.

You will need to write a program with a structured type, struct or class, perhaps called listnode, where each listnode contains an integral value field and a link. Here is a possible definition of that structure.

```
struct listnode;
typedef listnode*list;
struct listnode
 {
  int value;
  list link = NULL;
 };
```

I will provide data in form of a file containing a list of $n$ integers, which you will read into a linked list. During the input phase, you will need to construct $n$ dynamic variables type listnode, as well as the static variable of type list pointing to the first node in the list.

During this phase, you must create $n$ dynamic variables of type `listnode`, perhaps by using the constructor `new` $n$ times. After the first phase, you must not create any additional listnodes. Each listnode in your program will be created during the first phase, and its value will not be changed, although its link can be changed. You may create additional variables of type `list` as needed.

### Workspace.

At any given point during excecution, *workspace* is defined to be the memory needed to hold all variables other than the listnodes. This includes activation records for subprograms. Memory can be reused, and the number of pointers, activation records, and other variables at any given time must be $O(\log n)$. In

particular, this implies that the recursive depth of any subprogram must be $O(\log n)$. It also implies that you may not copy the values into an array.

**Format of the Data.**

Let $x_1, \ldots x_n$ be the list of integers. The input file has $n + 1$ lines. The first line is the value of $n$, the length of the list, the subsequent lines give the values. Here is an example:

```
4
300
121
415
225
```

The file **lst94** gives data for the final run of your program. You should use smaller files for debugging purposes.

Your program should write the original list and the sorted list: for example:

```
882 529 367 468 711 236 872 226 340 626 663 359 690 527 762 821 349 292 686 935
393 215 577 386 983
215 226 236 292 340 349 359 367 386 393 468 527 529 577 626 663 686 690 711 762
821 872 882 935 983
```

Hint: You will need subprograms. Some subprograms which may be useful are one which deletes the head of one linked list and prepends it to another, and another which which reverses the order of a linked list, such as:

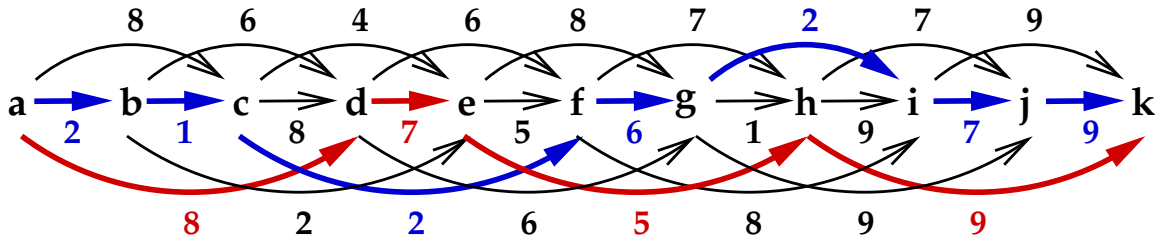
```
void shift(list&x,list&y)
 {
  .
  .
  .
 }

void reverselist(list&x)
 {
  .
  .
  .
 }
```

## Two Traveling Salesmen.

For this problem, you do not need to write a C++ program.

Let $G$ be a weighted acyclic directed graph, with a designated "start" node, $s$ and terminal node $t$. You wish to find two (directed) paths, each starting at $s$ and ending at $t$, such that every vertex of $G$ is on at least one of the two paths. The cost of a solution is the sum of the weights of the edges of the two paths. Design a dynamic programming algorithm that finds a solution of minimum cost, or which finds that no solution is possible. Remember: a path can't backtrack.

The figure below shows a weighted directed graph and a solution. The two paths are colored red and blue, respectively, and each has cost 29. That's a coincidence; the costs of the two paths can be different. In the example, the paths meet only at $s$ and $t$. But they may meet at any vertex.




Hint: The first step is to write the vertices in topological order. In the figure, the names of the vertices are letters, but for writing pseudocode, it is more convenient to given them numerical names.

## Traveler's Problem.

For this problem, write pseudocode, not an actual program.

A traveler must walk from $A$ to his home at $B$, staying at an inn during each night. There are $n$ inns on the road. The $i^{\text{th}}$ inn is $d_i$ miles from $A$ and costs $w_i$. The traveler can walk at most $d$ miles per day. How can he get home at minimum cost?

For simplicity, all numbers will be positive integers. Write an algorithm to find the minimum cost.

You may assume that $d_i < d_{i+1}$. The $n^{\text{th}}$ inn is the traveler's home and is located at $B$, hence $d_n$ is the distance from $A$ to $B$, and $w_n = 0$ since he doesn't have to pay to sleep at home.

Your algorithm should state whether there is a solution, and if so, should state give the minimum cost solution.

The time complexity of the obvious dynamic programming algorithm is $O(n^2)$. Can you do better? There is an $O(n)$ time algorithm for this problem.