Introduction to Programming

Another Sort of Sort

The following algorithm is often used to sort a hand of cards. We can represent the hand as a list of integers.

Let A be a list of integers and let i be an index into A (i.e., the position in A that we are examining). Here is an algorithm to sort A:

For each value of i starting at 1 and going to the length of A:

   Insert A[i] into the “correct” position in the sub-list to “left” of A[i], i.e., the sub-list A[0] – A[i-1].

By the “correct” position, we mean that position that will leave a[0] – a[i] sorted in ascending order.

In the example below, A[i] is underlined and it is moved “to the left” until it is in the correct position. For each line starting with the second, the underlined number is the one that will possibly be moving to the left. That number will be bolded on the next line so that you can see where it will end up. You can see that after each iteration i through the algorithm, the initial part of the list, the sub-list a[0] – a[i] will be sorted. When the algorithm terminates, the complete list A will be sorted.

3 7 4 9 5 2 6 1
3 7 4 9 5 2 6 1
3 7 4 9 5 2 6 1
3 4 7 9 5 2 6 1
3 4 7 9 5 2 6 1
3 4 5 7 9 2 6 1
2 3 4 5 7 9 6 1
2 3 4 5 6 7 9 1
1 2 3 4 5 6 7 9

Problem

See next Page.
Write a program to implement the sort described above according to the following requirements:

**Input**

1. Ask the user for an integer \( n \). If the user enters “done”, terminate the program.

2. Create a list \( A \) with \( n \) integers randomly generated from the range \([1,3*n]\). (Use function \texttt{randint(a,b)} that we saw in class.)

3. Print out the list you just generated.

**Sort the list**

Sort the list using the algorithm described above.

**Print the sorted list.**

**Go back to Input, above.**