Homework #4

Due: Thursday, November 13, 1997, at 11:59PM
Points: 100

The C Programming Language

Submitting: Put your program into a file called *main.c*, and submit it as described in the **All About Homework** handout. Remember, we'll compile and run it on the DECs, so be sure it works there! (**Note**: if you use another file name, we will deduct 5 points.)

The purpose of this question is for you to practise using pointers, dynamic storage allocation, and recursion.

Your program is to read input from the standard input. The input will be a series of lines, each with exactly one integer per line (there may be leading and/or trailing whitespace, but there will be only one number per line on well-formed input). Your program will insert each number into two different storage structures, one a linked list, and the other a binary tree. The numbers are to be inserted so that they are sorted. The program continues until there is no more input. It then prints the following data for each storage structure:

number of integers read
number of nodes in the storage structure
number of nodes visited to insert all the integers
mean number of nodes visited for each insertion
maximum number of nodes visited for any insertion

For example, suppose the input integers are 2, 5, 7, 3, 5, and 1 (in that order). Then your program would print

```
For the linked list:
6 integers read
5 nodes in the linked list
9 nodes visited to insert integers
1.5 nodes visited per insertion on the average
3 nodes visited in the worst case
For the binary tree:
6 integers read
5 nodes in the binary tree
8 nodes visited to insert integers
1.33 nodes visited per insertion on the average
2 nodes visited in the worst case
```

Your program must not print the sorted list unless an option –p is given; then the sorted list is to be printed **after** the above.

Notes: For the linked list, the list grows as: 2, 25, 257, 2357, 2357, 12357. To insert the integers, inserting 2 checks 0 nodes, 5 checks 1 node (the one containing 2). 7 checks 2 (2 and 5), 3 checks 2 (2 and 5), 5 checks 3 (2, 3, and 5), and 1 checks 1 (2). The total number of nodes checked is 0 + 1 + 2 + 2 + 3 + 1 = 9. That is 9/6 = 1.5 nodes visited per integer, on the average; and the most nodes checked to insert any single integer is 3 (to insert 5).

For the binary tree, inserting 2 checks 0 nodes, 5 checks 1 node (the one containing 2), 7 checks 2 nodes (2 and 5), 3 checks 2 (2 and 5), 5 checks 2 (2 and 5), and 1 checks 1 (2). The total number of nodes checked is 0 + 1 + 2 + 2 + 2 + 1 = 8. That is 8/6 = 1.33 nodes visited per integer, on the average; and the most nodes checked to insert any single integer is 2 (to insert 7, 3, and 5).