Programming Assignment 2 - Part II
List-Based Implementation quick sort
DUE: Sunday, Oct 8 by 6:00PM (24 hours before Exam-I).

You will write a program implementing a list-based version of quick-sort.

Key points:

- Your implementation must be written using the list ADT from programming assignment 1.
- You have been given a correct implementation of the list ADT.
- You have been given a list based implementation of the merge-sort algorithm for reference.
- You have been given code for doing input and output so you don't have to deal with that nitty-gritty stuff.

Program Behavior:

The given implementation of merge-sort (msort.c) and your implementation of quick-sort (qsort.c) behave in the same manner from the user perspective.

They simply read a sequence of integers from standard input. When the input is complete, the values are sorted using the appropriate algorithm and the resulting sorted sequence is printed to standard output.

Tips and Things to Remember:

- Your program MUST use the linked list ADT implementation from project-1. You should find that you have already done most of the work! These programs will be **clients** of the linked-list module.
- In case you are unsure of about the correctness of your linked-list functions from project-1, we have provided an implementation for your use/reference. You are free to use it instead of your implementation. We believe it to be bug-free.
- Since your program is a ***client*** of the linked list ADT, your code should NOT (and cannot) be fiddling around with the internals of the linked lists at all!
- Think about the fundamental operations QuickSort. How can they be achieved using the list primitives you've already written (merging two sorted sequence; partitioning about a pivot, etc.)

- Your sorting function should end up being short:
    - The given merge-sort function is about 15 lines long. Your quick-sort function should be of similar length.
- Your **pivot-selection strategy** for quick-sort does not need to be "smart". For purposes of this assignment, simply selecting the first element as the pivot is acceptable.
- Absolutely **no arrays** can be used -- directly or indirectly.
- Reading from standard input:

    The recommended method for reading the input is to repeatedly call scanf attempting to parse an integer each time until scanf fails or returns EOF.

    This also means that if the input contains 10 integers followed by some text, the program will sort the 10 integers and be done with it.

    Simple and sensible.

- You have been provided files io.c and io.h; you may use those functions to read the input and print/destroy the contents of a list. The read_ints() function complies with the specifications above. The program msort.c uses the provided io functions -- use it as an example.
- Submission: you will zip a directory containing all necessary source files and a makefile. When we unzip your directory we should be able to simply type the following to create your program.

```
> make qsort
```

Deliverables

You will zip your source files along with the Makefile described above and submit to Blackboard.
