Dana Vrajitoru
C243 Data Structures

Homework 9

Due date: Tuesday, November 18, 2014.

Ex. 1. This is a team homework.

Implement a sorting algorithm that is O(n $\log_2$ n) on average, such as Quicksort, Merge Sort, Heapsort. Your programs will the tested on a number of randmly generated arrays of various sizes (the same arrays for everyone). The fastest algorithm in the entire class turned in on time receives 5 extra credit points. Do not use the function sort from the STL, or any other function from the STL.

The main program should do the following, in this order:

- ask the user for the size of the array;
- allocate a dynamic array of the appropriate size;
- input all the elements of the array from the `cin` (I will use redirection from a file for the test);
- sort the array and time the sorting functions alone (see below);
- output the sorted array;
- output the timing information.

To test the algorithm you can generate the array using a random number generator, only change it before you turn the array in. This is designed so that we can use the Linux redirection to input the array from a file. Here is an example of a test file with 500 numbers to sort:
test500.txt

Timing. Here's some information about timing your sorting function. Note that this might not work just like this on Windows or Mac - you'd have to look for the appropriate header on that system. First, include this header file:

```
#include <sys/time.h>
```

Then declare the following variables:

```
struct timeval before, after;
double timing;
```

Before the sorting function:

```
gettimeofday(&before, 0);
```

After the function call:

```
gettimeofday(&after, 0);
timing = (double) ((double)after.tv_sec +
                   (double)after.tv_usec/(1000*1000)) -
         (double) ((double)before.tv_sec +
                    (double)before.tv_usec/(1000*1000));
```

The timing is your final timing of the sorting function.