Insertion Sort

```c
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>

//A basic insertion sort program that takes three command line arguments
//The first argument is the name of the file that contains the values to
//be sorted. The second is the number of elements in that file that will
//be sorted. The third argument is the file in which the program will
//print the results
int main( int argc, char *argv[] )  {
     int i, d, t;

     FILE *unsortedFile = fopen(argv[1], "r");
     FILE *sortedFile = fopen(argv[3], "w");

     char *numelts = argv[2];
     char *end;
     long count = strtol(numelts, &end, 10);
     int *nums=(int*)malloc(count*sizeof(int));

     for (i = 0; i < count; i++) {
        fscanf(unsortedFile, "%d\n", &nums[i]);
    }

    for (i = 1 ; i < count; i++) {
          d = i;
          while ( d > 0 && nums[d] < nums[d-1]) {
               t = nums[d];
               nums[d] = nums[d-1];
               nums[d-1] = t;
               d--;
          }
     }

    for (i = 0; i < count; i++) {
        fprintf(sortedFile, "%d\n", nums[i]);
    }

     free(nums);
    return 0;
}
```

Insertion sort is a basic sorting algorithm. The above code can be optimized in several ways. Most notably, it can achieve best case time complexity of $O(n)$ as opposed to the current best/average case being $O(n^2)$. There are other issues with this code as well which will be discussed in lab. However, insertion sort does have several desirable properties that are described at the beginning of its wikipedia page (e.g. stable, in-place, online, etc.).

Merge Sort

Mergesort is a much more widely used sorting algorithm than insertion sort, mainly because its average case time complexity is $O(n \log_2(n))$ which is far superior to insertion sort and other simple sorts. Mergesort is also stable, however it does not sort in-place. Therefore a drawback of mergesort is its memory use. The figure below gives a visual description of mergesort.