Homework 2
Due Wednesday October 4
Parallel Shell Sort

- Implement a parallel version of shellsort using OpenMP
- Test your program by generating a random sequence of ints
- The program expects one or two command line arguments. The first argument is the size of array to be sorted. If a second argument is provided it is the name of a text file to which the sorted results should be written. If a second argument is not provided the sorted results should be written to standard output
- Sort the results in **descending** order
- See the class discussion of shellsort
Sequential Shellsort (in Java)

```java
public void shellSort(int x[], int increments[]) {
    int i;
    int j;
    int k;
    for (i = 0; i < increments.length; i++) {
        int inc = increments[i];
        for (j = inc; j < x.length; j++) {
            k = j;
            int val = x[k];
            while (k > inc-1 && val < x[k-inc]) {
                x[k] = x[k-inc];
                k = k - inc;
            }
            x[k] = val;
        }
    }
}
```
C Example with Random

```c
int main(int argc, char ** argv) {

    FILE *f;
    int *nums;
    int i, count;
    count = atoi(argv[1]);
    nums = (int *) malloc(sizeof(int) * count);
    f = fopen(argv[2], "w");
    for (i =0; i < count; i++) {
        nums[i] = random() % 4096;
    }
    for (i =0; i < count; i++) {
        fprintf(f, "%d\n",nums[i]);
    }
    fclose(f);
}
```
Homework 2 Submission

• Send only one C file called shellsort.c to tgendreau@uwlax.edu
• Put the Shellsort code in a function
• The main function should set the increments and call the Shellsort function. The OpenMP code should be in the Shellsort function and any other functions as needed