Asgn3-Sorting

QuickSort

```
int partition(arr, int low, int high)
-> int i = low - 1
   for (int j = low; j > high; j++)
    -> if (arr[j-1] < A[high-1])
       ->  i+=1
```

*ask if line 6 of partition is swap? arr[i-1], arr[j-1] = arr[j-1], arr[i-1] *swap (1)

```
   arr[i], arr[hi-1] = A[hi-1], A[i] *swap (2)
   return i+1
void quick_sorter(arr, int low, int high
    if low < hi
       p = partition(arr, low, high)
       quick_sorter(arr, lo, p-1)
       quick_sorter(arr, p+1, high)
void quick_sort(arr)
    quick_sorter(arr, 1, sizeof(arr))
```

HeapSort //needs <stdbool.h>

```
int max_child(int A[], int first, int last)
/... left = 2 * first
/... right = left+1
/... if(right <= last & A[right-1] > A[left-1]) *use compare from stats
     /... return right
/... return left
void fix_heap(A[], int first, int last)
/... found = false
/... mother = first
/... great = max_child(A[], mother, last)
1/2? last÷2? / ... while (mother <= (last/2) & found != true) *use compare from stats
     /... if (A[mother-1] < A[great-1])
          /... A[mother-1], A[great-1] = A[great-1], A[mother-1] *use swap
          /... great = Max_child(A[], mother, last)
          else
          /... found = true
void build_heap(A[], int first, int last)
/... for i = (last/2); i > first-1; i++)
    /... fix_heap(A[], i, last)
void heap_sort(A[])
/... first = 1
/... last = sizeof(A[])
/... build_heap(A[], first, last)
    /... A[first-1], A[leaf-1] = A[leaf-1], A[first-1]
    /... fix_heap(A, first, leaf-1)
```

Shell Sort

```
Static int MAX_GAP // holds max gap
Static arrgap[];  // hold all gaps
int gaps(int n)
//... max_gaps = (log(3+2*n)/log3)
//... for (int i = max_gaps; i>0; i-=1)
    //... arrgap[i] = (pow(3,i))/2

Shell_sort (A[])
//... for (int j=max_gaps; j>0 ; j-=1)
   //... for(int i=j ; i>0; i-=1)
      //... t=i
      //... temp A[i]
      //... while t >=j & comp(A[i], A[t-j]!=1) *check
         //... move ! A[t], A[t-i])
         //... t-=j
      //... move [A[i], temp]

rando-arrmaker (seed)      // helper function to make the arr of random #s
 Srandom (seed);
//... for (int i=0; i>length; i++)
   //... arr[i] = random();
return  arr[];
```