sorting design doc

goal: Take in command arguments as a set, create an array of random numbers, sort with algorithms specified by command arguments and their set membership, and keep statistics of comparisons and swaps, printing out what was specified by command arguments at the end.

sudo:

sorting.c:
    main:
        parse command line args, insert into set
        randomize array, sort and print with selected algorithms.

quick.c:
    pos:
        set pivot
        for n-1
            arr val < pivot, swap
        swap i and n-1
        return i

    quick_sort:
        small = 8
        if n < small
            shell sort the list, return
        p = pos
        quick sort(arr, p)
        quick sort(arr + p + 1, n - p- 1)

batcher.c:
    comparator:
        swaps if two list elements aren't sorted

    batcher_sort:
        if length of list is 0 return
            k sort subsequences until k = 0

heap.c:
    l child return 2 * n + 1
    r child return 2 * n + 2
    parent return n-1 / 2

```
    up heap:
        while n > 0 and arr[n] < arr[parent(n)]
            swap a and parent
            n = parent

    down heap:
        n = 0
        while l child(n) < size
            if r child(n) = size temp = l child(n)
            else
                if l child(n) < r child(n) temp = l child(n)
            if a[n] < a[temp] break
            swap a[n] and a[temp]
            n = temp

    build heap:
        allocate heap, set all elements to 0
        copy data over to heap
        return heap

    heap_sort:
        allocate heap, set all elements to 0
        for n < elements
            new heap[n] = heap[0]
            heap[0] = heap[elements - n - 1]
            down heap
        copy new heap over to original list
        free temp heaps

shell.c:
    shell_sort:
        for gap in gaps
            for gap to length
                temp = arr[i]
                while i>=gap and temp < arr[i - gap]
                    arr[i] = arr[j-gap]
                arr[j] = temp
```