# CS 3: Introduction to Software Design

## Abstraction Exercises

**Name:**

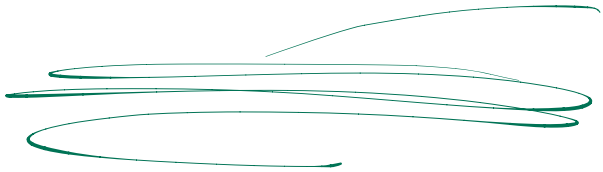
## Warmup

```
1  void sort(int *arr, int len) {
2     for (int i = 0; i < len; i++) {
3        int min_idx = [          ];
4        for (int j = i; j < len; j++) {
5           if (arr[j] < arr[min_idx]) {
6              min_idx = j;
7           }
8        }
9        swap(arr, [          ], [          ]);
10    }
11 }
```

```
1  /** Sorts the part of arr starting at lo (inclusive) and ending at hi (exclusive) **/
2  void sort_interval(int *arr, int lo, int hi) {
3
4
5
6
7
8
9
10
11
12
13
14 }
15 void sort(int *arr, int len) {
16
17
18
19 }
```

## Up and Down

```c
typedef int (*int_comparator_t)(const int a, const int b);
int int_asc(const int a, const int b);
int int_desc(const int a, const int b);
```



```c
/***
 * Sorts the part of arr starting at lo (inclusive) and ending at hi (exclusive)
 * according to the comparator `compare`
 ***/
void int_sort(int *arr, size_t lo, size_t hi, int_comparator_t compare) {



}

void ascending_int_sort(int *arr, size_t lo, size_t hi) {

}

void descending_int_sort(int *arr, size_t lo, size_t hi) {

}
```

## One More Time

```c
typedef int (*comparator_t)(const void *a, const void *b);
/* strcmp is a string comparator... */
int int_p_asc(const int *a, const int *b);
int int_p_desc(const int *a, const int *b);
```

strcmp

int *arr = {

};

sort(arr, 0, 5, int_asc);

↳ Can we cast btwn (int) and (void*)?

sizeof(int) ?= sizeof(void*)

void *a

int *i = malloc(...)

void *v = i;

```c
/***
 * Sorts the part of arr starting at lo (inclusive) and ending at hi (exclusive)
 * according to the comparator `compare`
 ***/
void sort(void **arr, size_t lo, size_t hi, comparator_t compare) {


}

void int_sort(int **arr, size_t lo, size_t hi, comparator_t compare)


}

void string_sort(char **arr, size_t lo, size_t hi, comparator_t compare) {

}
```

vec_list  vector_t ←

List_t  void * ← "object" "E"

sort(arr, lo, hi, strcmp)