CS136 EXERCISE SOLUTIONS: MODULE 5

Arrays

```
int int_sum(int a[], int len) {
  int total = 0;
  for (int i = 0; i < len; ++i) {
    total += a[i];
  }
  return total;
}
```

```
void array_map(int (*f)(int), int a[], int len) {
  for (int i = 0; i < len; ++i) {
    a[i] = f(a[i]);
  }
}
```

Pointer arithmetic

```
int char_sum(char *p, int count) {
  int total = 0;
  while (count > 0) {
    total += *p;
    ++p;
    count--;
  }
  return total;
}
```

An array is a lot like a pointer

`<something> f(int *arr)` or equivalentlxy: `<something> f(int arr[])`

Selection sort

```
void selection_sort(int a[], int len) {
  for(int i=0; i<len; ++i) {
    print_array(a, 7);

    int index_smallest = i;
    for(int j = i+1; j<len; ++j) {
      if (a[j] < a[index_smallest]) {
        index_smallest = j;
      }
    }
    swap(&a[i], &a[index_smallest]);
  }
}
```

Binary search

```
int binary_search(const int target, const int a[], const int len) {
  int left = 0;
  int right = len - 1;
  while (left <= right) {
    int mid = (left + right) / 2;
    if (a[mid] == target) {       // found it!
      return mid;
      // L----M----R
    } else if (a[mid] > target) { // It's in the left side, so
      right = mid - 1;            //   move our right edge toward the left.
      // L---R......
    } else {                      // It's in the right side, so
      left = mid + 1;             //   move our left edge toward the right.
      // ......L---R
    }
  }
  return -1; // There are no slots left and we didn't find it.
}
```

```
// print_matrix(a, width, height) Print all values in a, when viewed
// as a matrix of given width and height.
// requires: width and height > 0
void print_matrix(const int a[], const int width, const int height) {
  assert(a != NULL);
  assert(width > 0);
  assert(height > 0);
  for (int rown = 0; rown < height; ++rown) {
    printf("[ ");
    for (int columnn = 0; columnn < width; ++columnn) {
      printf("%d ", a[rown * width + columnn]);
    }
    printf("]\n");
  }
```