Lab 4

You are expected to copy and paste your code into each corresponding box in this handout and submit it as a Word document or PDF file before the due. Additionally, your lab instructor will tell you which three questions you must showcase during the lab session. While you may demonstrate your code running in person after the due date, your file must be submitted on time.

Hint #1: For questions asking for 'a **function** which takes in xyz', the xyz refer to function parameters, not the use of scanf().

Hint #2: For questions asking for 'a **program** which reads in xyz', the xyz refer to the use of scanf().

Hint #3: For questions asking for 'a program', you are expected to include all necessary #include and define your main function. It's up to you whether to define additional helper functions, though usually, you don't need to.

Task 1: Write a **program** which defines an int array with length 10, gives it some initial values, and then prints out the values from the array along with each value's index

```
#include <stdio.h>
#define LENGTH 10

int main(void) {
    int numbers[LENGTH] = {1, 1, 2, 3, 5, 8, 13, 21, 34,
                           55};

    for (int i = 0; i < LENGTH; ++i) {
        printf("%i %i\n", i, numbers[i]);
    }

    return 0;
}
```

Task 2: Write a program which reads in a string name, Eg, “Tom”, and then prints out a greeting “Hello Tom!”

```
#include <stdio.h>

int main(void) {
     char name[100];
     printf("Enter a name: \n");
     scanf("%s", name);

     printf("Hello %s!\n", name);

     return 0;
}
```

Task 3: Write a program which reads in a string from the user, and then prints a version without the first and last char, so for "Hello" prints "ell".
The string length will be at least 2.

input: Hello → print: ell
input: salute → print: alut
input: coding → print: odin

```
#include <stdio.h>

int main(void) {
     char string[100];
     printf("Enter a string (length < 100): \n");
     scanf("%s", string);

     for (size_t i = 0; string[i] != '\0'; ++i) {
         if (string[i + 1] == '\0') {
              string[i] = '\0';
         }
     }

     printf("%s\n", string + 1);

     return 0;
}
```

Task 4: Write a **function** (named “nonStart”) which takes in two char arrays, and then prints their concatenation, except omit the first char of each. The strings will be at least length 1. Eg,

nonStart("Hello", "There") → print: "ellohere"
nonStart("java", "code") → print: "avaode"
nonStart("shotl", "java") → print:"hotlava"

```
void nonStart(char first[], char second[]) {
     printf("%s", first + 1);
     printf("%s\n", second + 1);
}
```

Task 5: Write a **function** which takes in an input integer array and the length of the array. The function is to reverse the array values. Eg, input [1, 2, 3] -> [3, 2,1]
[Hint: This function should not print, it changes values in the given array.]

```
void reverse(int arr[], size_t length) {
     size_t lo = 0, hi = length - 1;

     while (lo < hi) {
          int hold = arr[lo];
          arr[lo] = arr[hi];
          arr[hi] = hold;
          ++lo; --hi;
     }
}
```

Task 6: Write a **function** which takes in an int array and the length of it, finds and returns the mean of the array
Prototype: **double mean (const int data[], size_t length);**

```
double mean(int arr[], size_t length) {
     if (length <= 0) { return 0; }

     double sumSoFar = 0;
     for (size_t i = 0; i < length; ++i) {
         sumSoFar += arr[i];
     }

     return sumSoFar / length;
}
```

Task 7: Write a **function** which takes in an int array and the length of it, finds and returns one number from the array which has the biggest absolute value.
Prototype: **int max_abs (const int data[], size_t length);**

```
int max_abs(int arr[], size_t length) {
     int max_so_far = -1;
     for (size_t i = 0; i < length; ++i) {
     int abs_element = arr[i] > 0 ? arr[i] : -arr[i];
         if (abs_element > max_so_far) {
              max_so_far = abs_element;
         }
     }
     return max_so_far;
}
```

Task 8: Write a **function** which takes in an int array and the length of it, along with two other integers val_1 and val_2, finds and returns how many numbers in the array is between val_1 and val_2, inclusively.
[Hint: to determine how many numbers are val_1 ≤ number ≤ val_2]
Prototype:
size_t in_between (const int data[], size_t length, int val_1, int val_2);

```
size_t in_between(int arr[], size_t length, int val_1,
                  int val_2) {
    size_t num = 0;
    for (size_t i = 0; i < length; ++i) {
        int element = arr[i];
        if (((val_1 <= element) && (element <= val_2)) ||
            ((val_1 >= element) && (element >= val_2))) {
            ++num;
        }
    }
    return num;
}
```

Task 9: Write a **function** which takes in a two-dimensional int array (with size of n-by-10) and an integer n for the row number, to find and return the minimum value in the array [Hint: column number is 10]
Prototype: **int minimum (const int data [][10], const size_t n);**

```
int minimum(const int data[][10], const size_t n) {
    int min = 2147483647; // INT_MAX
    for (size_t i = 0; i < 10; ++i) {
        for (size_t j = 0; j < n; ++j) {
            if (data[j][i] < min) {
                min = data[j][i];
            }
        }
    }
    return min;
}
```

Task 10: Write a **function** which takes in a two-dimensional int array (with size of n-by-7) and integer n for the row number, to find and return how many even numbers are in the array [Hint: column number is 7]
Prototype: **size_t count_even (const int data [][7], const size_t n);**

```
size_t count_even(const int data[][7], const size_t n) {
     size_t even_count = 0;
     for (size_t i = 0; i < 7; ++i) {
         for (size_t j = 0; j < n; ++j) {
              if (!(data[j][i] % 2)) {
                   ++even_count;
              }
         }
     }
     return even_count;
}
```