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.

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>  int main() {  int numbers[10] = {5, 12, 7, 22, 15, 3, 9, 18, 1, 6};  printf("Index\tValue\n");  printf("------------------\n");   for (size\_t counter = 0; counter < 10; counter++) {  printf("%llu\t%d\n", counter, numbers[counter]);  }  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() {  char name[20];  printf("Enter your name:\n");  scanf("%19s", name);  printf("Hello, %s!", 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() {  char input[20];  printf("Enter a string value:\n");  scanf("%19s", input);   size\_t length = 0;  while (input[length] != '\0') {  length++;  }   printf("Truncated result: ");  for (size\_t index = 1; index < (length - 1); index++) {  printf("%c", input[index]);  }  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 string1[], char string2[]) {  size\_t index;   // String 1  for (index = 1; string1[index] != 0; index++) {  printf("%c", string1[index]);  }  // String 2  for (index = 1; string2[index] != 0; index++) {  printf("%c", string2[index]);  }  printf("\n"); } |

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 reverseIntegers(int input[], size\_t length) {  for (size\_t index = 0; index < (length / 2); index++) {  int temp = input[index];  input[index] = input[length - 1 - index];  input[length - 1 - index] = temp;  } } |

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(const int data[], size\_t length) {  int sum = 0;  for (size\_t index = 0; index < length; index++) {  sum += data[index];  }  return (sum / length) \* 1.0; } |

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);**

|  |
| --- |
| #include <stdlib.h>  int max\_abs(const int data[], size\_t length) {  int maxValue = abs(data[0]);  for (size\_t index = 1; index < length; index++) {  if (abs(data[index]) > abs(maxValue))  maxValue = abs(data[index]);  }  return maxValue; } |

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(const int data[], size\_t length, int val\_1, int val\_2) {  size\_t count = 0;   if (val\_1 > val\_2) {  int temp = val\_1;  val\_1 = val\_2;  val\_2 = temp;  }  for (size\_t index = 0; index < length; index++) {  if (data[index] >= val\_1 && data[index] <= val\_2) {  count++;  }  }  return count; } |

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 minValue = data[0][0];   for (size\_t indexRow = 0; indexRow < n; indexRow++) {  for (size\_t indexCol = 0; indexCol < 10; indexCol++) {  if (data[indexRow][indexCol] < minValue) {  minValue = data[indexRow][indexCol];  }  }  }  return minValue; } |

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 count = 0;   for (size\_t indexRow = 0; indexRow < n; indexRow++) {  for (size\_t indexCol = 0; indexCol < 7; indexCol++) {  if (data[indexRow][indexCol] % 2 == 0) {  count++;  }  }  }  return count; } |