Computer Engineering Department

TED University

TED UNIVERSITY

CMPE 252 - C Programming, Spring 2023

Lab 2

Part I (30 points)

In this part, you will write a program which involves implementation of the following two functions.

```
void readInput(int arr[], int *nPtr); // reads numbers from the standard input into arr, and stores the number of elements read in the memory cell pointed to by nPtr
```

```
void printNumbers(const int arr[], int n); // prints the elements in arr[0..(n-1)]
```

First, define a constant macro named `SIZE` with the value `1000`.

In main function, you will create an array and print the elements of the array as follows:

- Define an integer array with the size `SIZE`
- Call `readInput` function
- In the `readInput` function,
  - First, read number of elements into the memory cell pointed by `nPtr`.
  - Then, read elements into `arr`.
- Call `printNumbers` function for printing the array elements.

Sample Run:

```
Enter the number of elements:
5
Enter 5 elements:
1 2 3 4 5
Array elements: 1 2 3 4 5
```

Part II (35 points)

Your task in this part to fill in the missing function definitions in skeleton code **lab2part2.c**. You will use the same `readInput` and `printNumbers` functions from part I. **main** function will stay as it is.

Implement the following function in skeleton code **lab2part2.c**:

```
// Precondition: Let n represent number of elements in arr.
/* Finds mean value of arr and stores it in the memory cell pointed to by meanPtr. (Mean = average of the numbers) */
/* Finds variance value of arr and stores it in the memory cell pointed to by varPtr. (Variance = (summation ((arr_i - average of numbers) * (arr_i - average
```

```
of numbers))) / Total no of elements. where i = 1 to n here n is the total no
of elements.) */
void findMeanVar(const int arr[], int n, double *meanPtr, double *varPtr);
```

Here are the formulas for mean and variance:

$$mean = \left(\sum_{i=0}^{n-1} a[i] \right) / n$$

$$variance = \left(\sum_{i=0}^{n-1} (a[i] - mean)^2 \right) / n$$

Sample Run:

```
Enter the number of elements:
9
Enter 9 elements:
1 2 3 4 5 6 7 8 9
Array elements: 1 2 3 4 5 6 7 8 9
Mean of all elements = 5.00
Variance of all elements = 6.67
```

Part III (35 points)

Your task in this part to fill in the missing function definitions in skeleton code **lab2part3.c**. You will use the same `readInput` and `printNumbers` functions from part I. **main** function will stay as it is.

Implement the following function in skeleton code **lab2part3.c**:

```
// Precondition: Let n represent size of unsorted arr of distinct integers.
/* Fill in csmallerArr which includes the count of smaller elements on the
right side of each element in the arr. */
void countSmallerArray(const int arr[], int n, int csmallerArr[]);
```

Sample Run:

```
Enter the number of elements:
7
Enter 7 elements:
12 1 2 3 0 11 4
Array elements: 12 1 2 3 0 11 4
Count Smaller Array elements: 6 1 1 1 0 1 0
```