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 the count of negative elements in the arr and stores in the memory cell pointed to by negCountPtr.
// Finds the count of non-negative elements in the arr and stores in the memory cell pointed to by nonnegCountPtr.
void countNegNonneg(const int arr[], int n, int *negCountPtr, int *nonnegCountPtr);
```

Sample Run:

```
Enter the number of elements:
10
Enter 10 elements:
1 -2 3 -4 5 -6 -7 8 9 10
Array elements: 1  -2  3  -4  5  -6  -7  8  9  10
Count of Non-negative elements = 6
Count of Negative elements = 4
```

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 number of elements in arr.
/* Finds all loosers in arr and stores into loosersArr and number of elements
in loosersArr is stored in the memory cell pointed to by sp. */
/* An element is a looser if it is smaller than all the elements to its left
side. And the leftmost element is always a looser. */
void findLoosers(const int arr[], int n, int loosersArr[], int *sp)
```

Sample Run:

```
Enter the number of elements:
6
Enter 6 elements:
6 7 4 3 5 2
Array elements: 6 7 4 3 5 2
Loosers Array elements: 6 4 3 2
```