Data Structures & Algorithms LAB
(BSCS-F18 Morning & Afternoon)
Lab # 2

Instructions:

- **Make sure that there are no dangling pointers or memory leaks in your programs.**
- **Indent your code properly**
- **Use meaningful variables and function names.**
- **Use meaningful prompt lines/labels for all input/output.**

Task # 1
You are given two arrays of integers, both containing exactly **n** integers. Write 3 different functions to determine the **intersection** (i.e. common elements) of these two arrays, as described below. Assume that there are **no duplicates** in either of the two arrays.

1.1 Implement the function:

int intersection1 (int* A, int* B, int* C, int n)

Which takes three integer arrays (each array having size/capacity of **n**) as parameters. The intersection of arrays **A** and **B** will be stored in the array **C**. Note that the previous contents stored in the array **C** will be over-written by this function. This function should also return the number of elements that were stored in the array **C**.
The worst-case time complexity of this function should be $O(n^2)$. You are NOT allowed to allocate any new array within this function.

1.2 Now, you can assume that both the input arrays (**A** and **B**) are **sorted** in **increasing order**. Implement the following function to determine the intersection of the two arrays:

int intersection2 (int* A, int* B, int* C, int n)

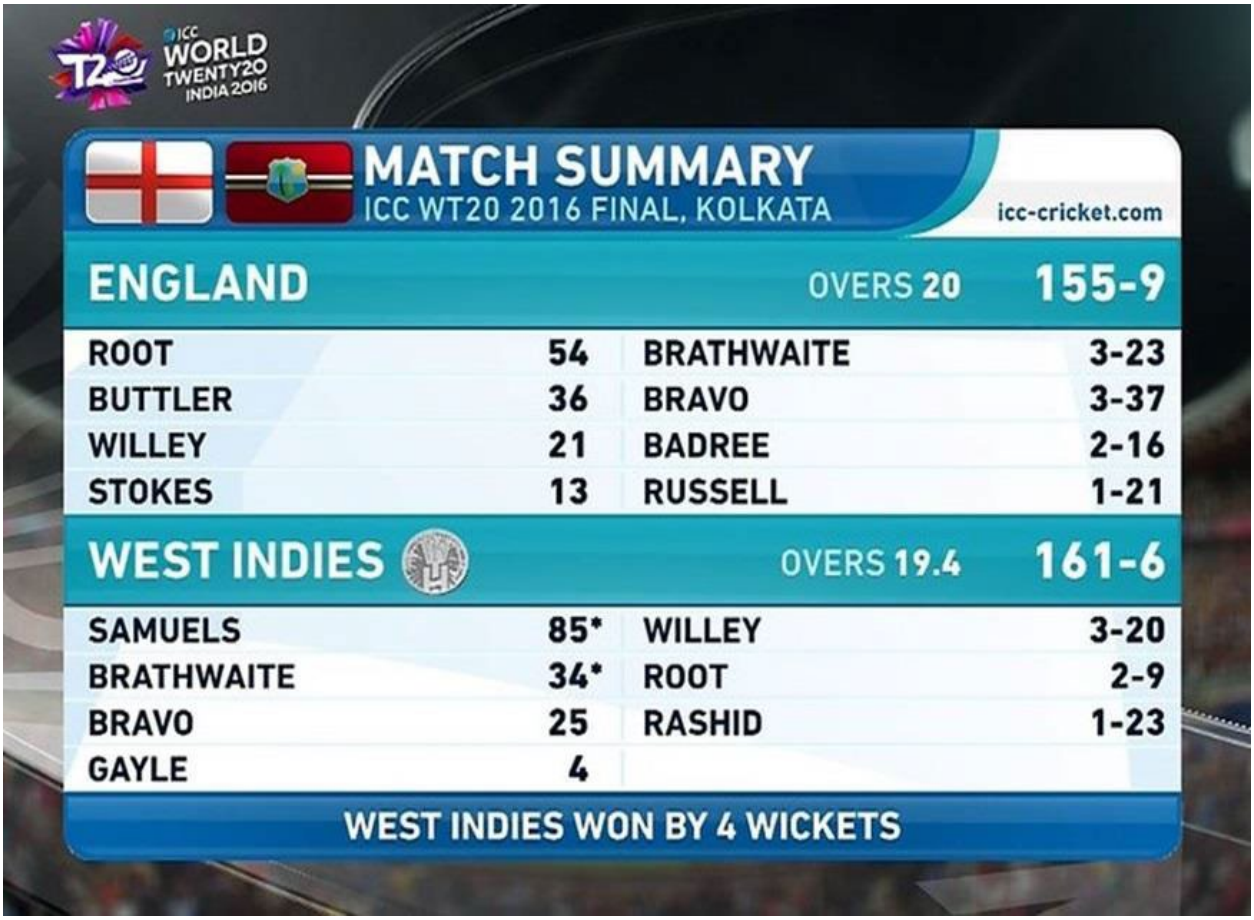
The worst-case time complexity of this function should be $O(n \lg n)$. You are NOT allowed to allocate any new array within this function.

Write a main function which illustrates the working of the above 2 functions.

Task # 2

In the match summary, which is displayed at the end of a cricket match, bowling figures of bowlers are displayed in **decreasing order of Wickets** that they have taken. Furthermore, if two (or more) bowlers have the same number of wickets, then the bowler conceding **lesser runs** is

displayed first. As an example, see the following match summary of the final match of the ICC WT20 2016:



In the first inning, you can see that the bowling figures of West Indian bowlers are displayed in decreasing order of wickets. There are two bowlers which have taken 3 wickets, so there bowling figures have been displayed in increasing order of runs that they have conceded.In this task, you are required to sort a list of bowling figures. You are required to use the following class**BowlingFigures**in your implementation:

```
constint NAME_LENGTH = 25;

classBowlingFigures
{
      char name[NAME_LENGTH+1]; // Name of the bowler
      int wickets; // Wickets taken by the bowler
      int runs; // Runs conceded by the bowler
};
```

Your program should take its input from a text file. A sample input file1 is shown below:

```
AamerSohail
0 49
AaqibJaved
2 27
WasimAkram
3 49
Ijaz Ahmed
0 13
Mushtaq Ahmed
3 41
Imran Khan
1 43
```

Each bowler's bowling figure is on two lines in the input file. The first line contains the name of the bowler, while there are two integers on the second line which indicate the number of wickets taken and the runs conceded by the bowler, respectively. You can assume that the name of a player will contain 25 characters at max.

You are required to implement the following 3 functions:

```
BowlingFigures* readFromFile (char* fileName, int& count);
```

This function will take the name of an input file and an un-initialized integer variable as parameters. If the input file is not found, this function should return NULL. If the file is opened successfully, this function will dynamically allocate an array of **BowlingFigures**and read all the bowling figures into this array. This function should store the count of the No. of bowling figures into the reference parameter (**count**).

```
voiddisplayBowlingFigures (BowlingFigures * bp, int count);
```

This function will take the array of **BowlingFigures**and its size as parameters, and it will display all the bowling figures on screen, in a neat and readable way.

```
voidsortBowlingFigures (BowlingFigures * bp, int count);
```

This function will take the array of **BowlingFigures**and its size as parameters, and it will sort this array into **decreasing order of bowling figures** i.e. the bowling figures must be arranged in decreasing order of wickets taken, and if two (or more) bowlers have taken the same number of wickets, then the bowler conceding lesser number of runs should be displayed first. You MUST use **Selection Sort** to sort the array of bowling figures. You can assume that in the input file, no two bowling figures will be exactly identical.

For the input file shown on the previous page, a sample run of your program will produce the following output:

```
Following 6 Bowling Figures were read from the input
file:
      AamerSohail 0-49
      AaqibJaved 2-27
      WasimAkram 3-49
      Ijaz Ahmed 0-13
      Mushtaq Ahmed 3-41
      Imran Khan 1-43
Bowling figures after sorting are:

      Mushtaq Ahmed 3-41
      WasimAkram 3-49
      AaqibJaved 2-27
      Imran Khan 1-43
      Ijaz Ahmed 0-13
      AamerSohail 0-49
```

Task # 3

The history teacher at your school needs help in grading a True/False test. The students' IDs and test answers are stored in a file ("input.txt"). The first entry in the file contains answers to the test in the form:

TFFTFFTTTTFFTFTFTFTT

Every other entry in the file is the student ID, followed by a blank, followed by the student's responses. For example, the entry:

BCSF12A001 TFTFTFTT TFTFTFFTTFT

indicates that the student ID is BCSF12A001 and the answer to question 1 is True, the answer to question 2 is False, and so on. This student did not answer question 9. The exam has 20 questions, and the class has more than 10 students. Each correct answer is awarded two points, each wrong answer gets one point deducted, and no answer gets zero points. Write a program that processes the test data. The output should be the student's ID, followed by the answers, followed by the test score, followed by the test grade. Assume the following grade scale: 90%–100%, A; 80%–89.99%, B; 70%–79.99%, C; 60%–69.99%, D; and 0%–59.99%, F.