Programming Fundamentals *COURSE & LAB* – Spring 2022

(NC-BS-IT-F21 Morning)

Assignment# 03

*Assigned on: **Wednesday, November 16, 2022***

*Submission Deadline of assignment questions: **Wednesday, November 23, 2022 (till 11:59 PM)***

Instructions:

- **This is an individual assignment. Absolutely NO collaboration is allowed. Any traces of plagiarism/cheating would result in an "F" grade in this course.**
- Do **NOT** copy even a single line of code from any other person or book or Internet or any other source.
- This assignment will ALSO be counted towards your marks in **PF-Lab**.
- Late submissions will NOT be accepted, whatever your excuse may be.
- This assignment needs to be submitted in **Soft** form. The **Submission Procedure** is given on last page.
- Clearly mention your **Name**, **Roll Number** and **Section** in comments at the **top** of **each CPP file**.

Question # 1 — 10 Marks

Define a C function having four parameters. This function should take an array of integers and its size as its first two parameters. This function should determine the most frequent element of the array, and it should return the **most frequent element** (i.e. the element which occurs the most often) and its **frequency** through its *last two reference parameters*. In your function, you are NOT allowed to declare or use any other array. Illustrate the working of your function (on various inputs) in a complete C program. See the following examples:

Example #	Elements of the array	Most frequent element	Frequency
1	**5 3 4 5 2 7 5 4**	**5**	**3**
2	**12 74 23 35 23 12**	**12** or **23**	**2**
3	**7 5 6**	**7** or **5** or **6**	**1**

As you can see from the last two examples, if more than one element is the most frequent element, then your function can determine any one of them as the most frequent element.

Illustrate the working of your function (on at least 5 different examples/inputs) in a complete C program.

Question # 2 10 Marks

Define a C function that should remove all NEGATIVE values from an unsorted array of integers. Your function should also preserve the order of remaining elements in the array. In the end, this function should also return the count of the NEGATIVE values that were removed from the array. The prototype of your function should be:

```
int removeNegatives (int arr[], int size, int *newSize)
```

Note that the third reference parameter is used to send the updated size of the array back to the calling function.

For example, if **arr** contains these **7** elements **{ 11, -15, -2, 7, 11, 6, -8 }**, then, after the function call to **removeNegatives** the **arr** should now contain **{ 11, 7, 11, 6 }** in its first four indices. The variable **newSize** (see the 3rd parameter) should contain **4** (since the *partially filled array* **arr** now contains 4 valid elements) and the function should return **3** (because 3 negative values were removed from the array).

Important Note:

- In your function, you are NOT allowed to declare or use any other array.

Illustrate the working of your function (on at least 4 different examples/inputs) in a complete C program.

Question # 3 10 Marks

Define a C function:

```
void cyclicRotate (int arr [], int n, int k);
```

This function takes an array (**arr**) and its size (**n**) as its first two parameters. Third parameter is an integer value **k**. This function should *cyclically rotate* all the elements of the array **arr**, by **k** positions to the right.

For example,

if **arr** contains these **8** elements **{ 3, 4, 5, 8, 7, 2, 9, 1 }**,

then, after the function call **cyclicRotate (arr, 8, 3)**

the **arr** should contain these **8** elements **{ 2, 9, 1, 3, 4, 5, 8, 7 }**.

In your function:

- You can assume that **k** is greater than **0** and less than **n**
- You are NOT allowed to declare or use any other array.

Illustrate the working of your function (on at least 4 different examples/inputs) in a complete C program.

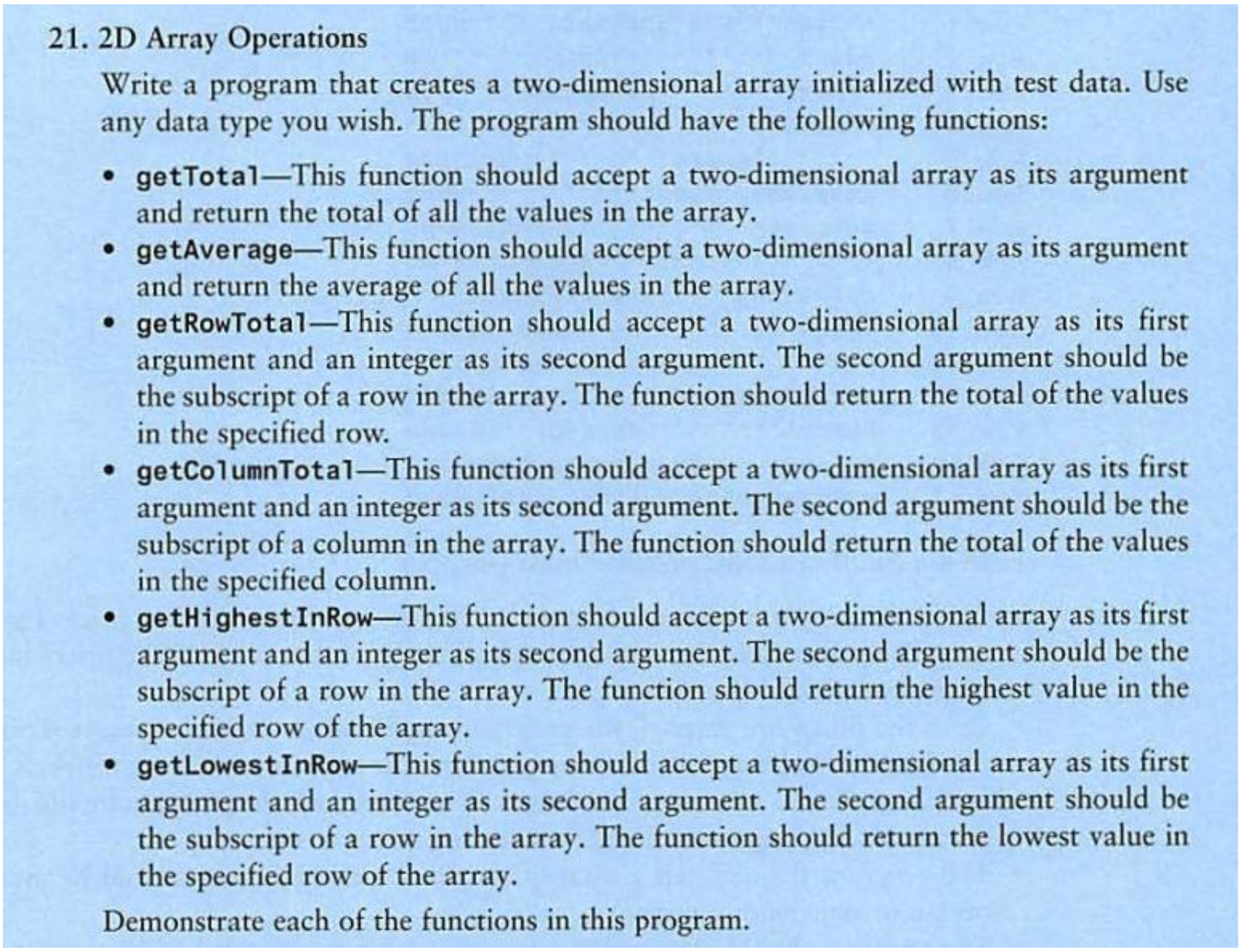
Hint 1: What happens if **`k`** is **`1`**?

Hint 2: You may implement another helper function to accomplish this task.

Hint 3: Do some paperwork.

Question # 4 — 10 Marks

Write a C program for solving **Programming Challenge #21** (*2D array operation*) on **Page 460** of your textbook (Gaddis, 9th Edition).



21. 2D Array Operations

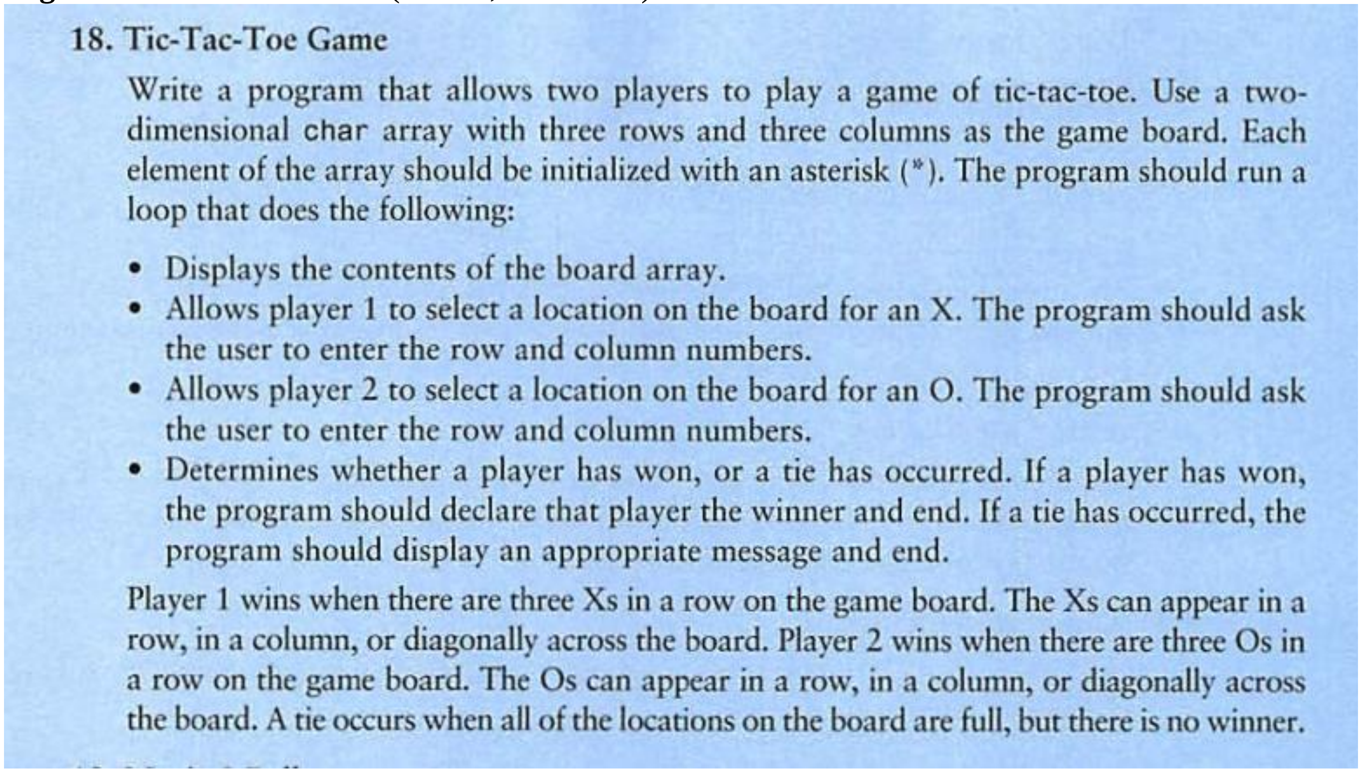
Write a program that creates a two-dimensional array initialized with test data. Use any data type you wish. The program should have the following functions:

- **`getTotal`**—This function should accept a two-dimensional array as its argument and return the total of all the values in the array.
- **`getAverage`**—This function should accept a two-dimensional array as its argument and return the average of all the values in the array.
- **`getRowTotal`**—This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the total of the values in the specified row.
- **`getColumnTotal`**—This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a column in the array. The function should return the total of the values in the specified column.
- **`getHighestInRow`**—This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the highest value in the specified row of the array.
- **`getLowestInRow`**—This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the lowest value in the specified row of the array.

Demonstrate each of the functions in this program.

Question # 5 20 Marks

Write a C program for solving **Programming Challenge # 18** (*Tic-Tac-Toe Game*) on **Page 460** of a reference book (Gaddis, 9th Edition).



18. Tic-Tac-Toe Game

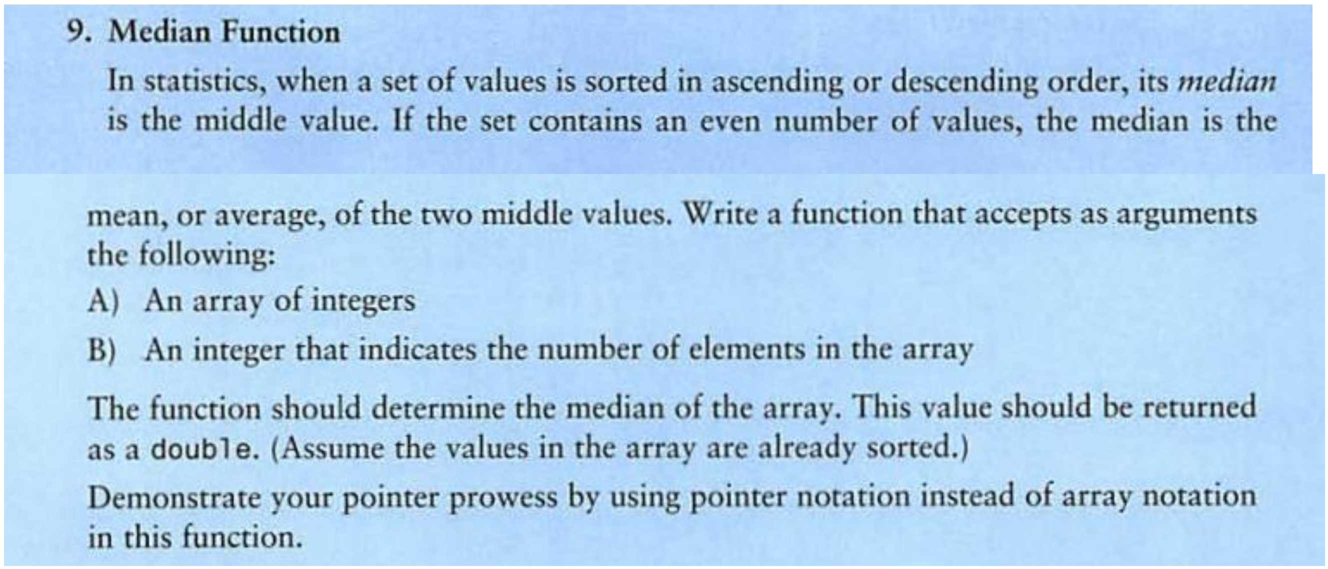
Write a program that allows two players to play a game of tic-tac-toe. Use a two-dimensional `char` array with three rows and three columns as the game board. Each element of the array should be initialized with an asterisk (*). The program should run a loop that does the following:

- Displays the contents of the board array.
- Allows player 1 to select a location on the board for an X. The program should ask the user to enter the row and column numbers.
- Allows player 2 to select a location on the board for an O. The program should ask the user to enter the row and column numbers.
- Determines whether a player has won, or a tie has occurred. If a player has won, the program should declare that player the winner and end. If a tie has occurred, the program should display an appropriate message and end.

Player 1 wins when there are three Xs in a row on the game board. The Xs can appear in a row, in a column, or diagonally across the board. Player 2 wins when there are three Os in a row on the game board. The Os can appear in a row, in a column, or diagonally across the board. A tie occurs when all of the locations on the board are full, but there is no winner.

Question # 6 10 Marks

[Pointers]Write a C program for solving **Programming Challenge # 09** (*Theater Seating*) on **Pages 544-545** of a reference book(Gaddis, 9th Edition).



9. Median Function

In statistics, when a set of values is sorted in ascending or descending order, its *median* is the middle value. If the set contains an even number of values, the median is the mean, or average, of the two middle values. Write a function that accepts as arguments the following:

A) An array of integers

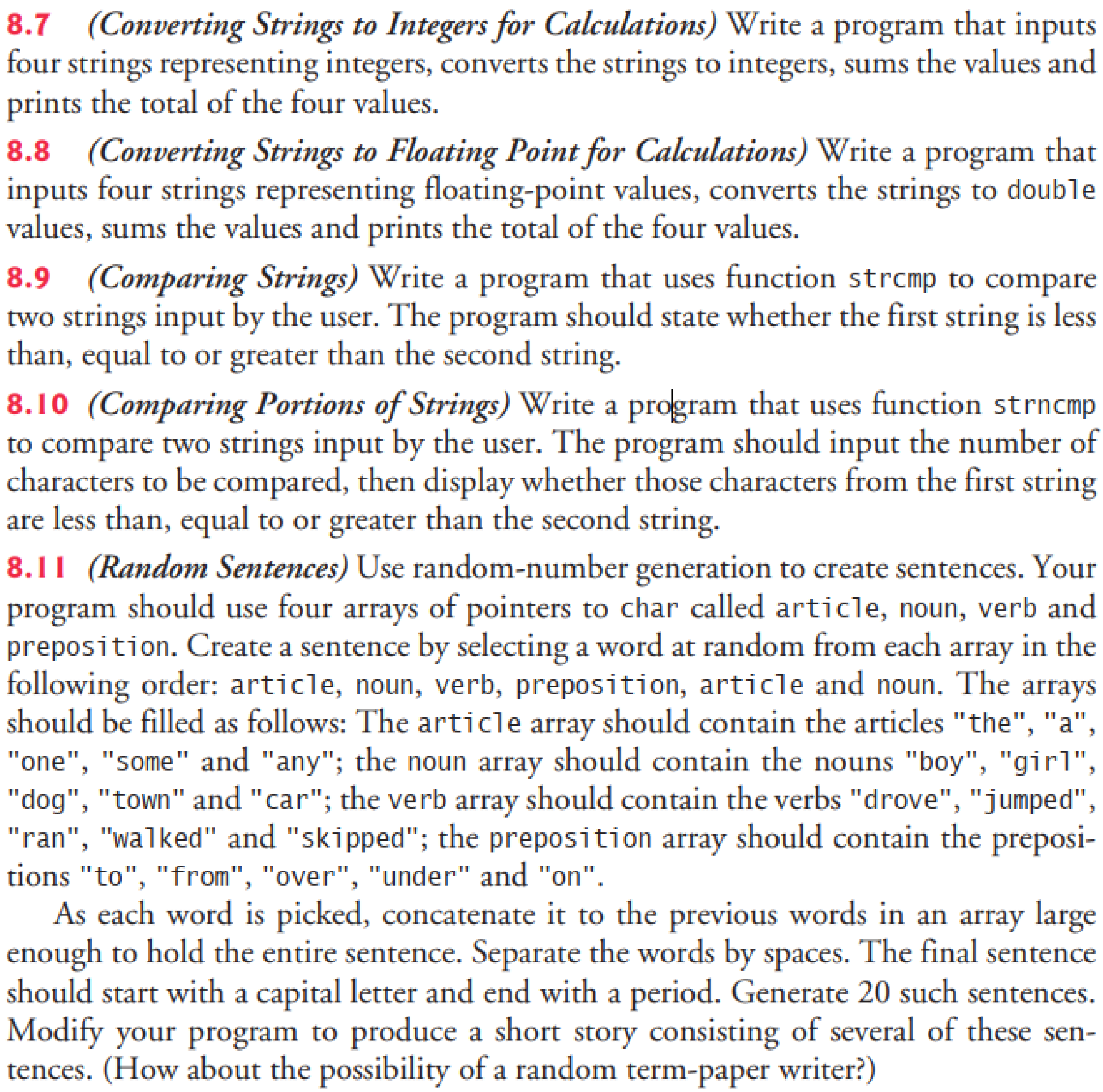
B) An integer that indicates the number of elements in the array

The function should determine the median of the array. This value should be returned as a `double`. (Assume the values in the array are already sorted.)

Demonstrate your pointer prowess by using pointer notation instead of array notation in this function.

Question # 7 5+5+5+5+10= 30 Marks

[Strings handling] Write a C program for solving **Programming questions 8.7 to 8.11** on **Pages 427** of your textbook (Deitel, 9th Edition).



8.7 *(Converting Strings to Integers for Calculations)* Write a program that inputs four strings representing integers, converts the strings to integers, sums the values and prints the total of the four values.

8.8 *(Converting Strings to Floating Point for Calculations)* Write a program that inputs four strings representing floating-point values, converts the strings to `double` values, sums the values and prints the total of the four values.

8.9 *(Comparing Strings)* Write a program that uses function `strcmp` to compare two strings input by the user. The program should state whether the first string is less than, equal to or greater than the second string.

8.10 *(Comparing Portions of Strings)* Write a program that uses function `strncmp` to compare two strings input by the user. The program should input the number of characters to be compared, then display whether those characters from the first string are less than, equal to or greater than the second string.

8.11 *(Random Sentences)* Use random-number generation to create sentences. Your program should use four arrays of pointers to `char` called `article`, `noun`, `verb` and `preposition`. Create a sentence by selecting a word at random from each array in the following order: `article`, `noun`, `verb`, `preposition`, `article` and `noun`. The arrays should be filled as follows: The `article` array should contain the articles `"the"`, `"a"`, `"one"`, `"some"` and `"any"`; the `noun` array should contain the nouns `"boy"`, `"girl"`, `"dog"`, `"town"` and `"car"`; the `verb` array should contain the verbs `"drove"`, `"jumped"`, `"ran"`, `"walked"` and `"skipped"`; the `preposition` array should contain the prepositions `"to"`, `"from"`, `"over"`, `"under"` and `"on"`.

As each word is picked, concatenate it to the previous words in an array large enough to hold the entire sentence. Separate the words by spaces. The final sentence should start with a capital letter and end with a period. Generate 20 such sentences. Modify your program to produce a short story consisting of several of these sentences. (How about the possibility of a random term-paper writer?)

Submission Procedure:

i. Now, compress the folder (that you created in previous step) in **.RAR** or **.ZIP** format. The name of the RAR or ZIP file should be according to format:

`Mor-A3-BITF20M123` (for Morning section) OR

`Aft-A3-BITF20A456` (for Afternoon section),

where the text shown in **BLUE** should be your **complete Roll Number**.

ii. Finally, submit the (single) **.RAR** or **.ZIP** file through **Google Classroom**. Make sure that you press the ***SUBMIT*** button after uploading your file.

Note: *If you do not follow the above submission procedure, your Assignment will NOT be graded and you will be awarded ZERO marks.*

These *good programming practices* will also have their (significant) weightage in the markingof this assignment:

- Comment your code intelligently. **Uncommented code will NOT be given any credit.**
- Indent your code properly.
- Use meaningful variable names.
- ○ Use meaningful prompt lines/labels for input/output.
- If any program that you submit **does NOT compile correctly** or it **generates any kind of run-time error** you will get ZERO marks in the whole assignment.
-

☺ GOOD LUCK! ☺