

COMSC 165

Spring 2020

Programming Assignment 8

Worth 20 points (2% of your grade)

**DUE: Wednesday, 7/15/20 by 11:59 P.M. on
Canvas**

START by downloading the 165_assign8.cpp file from Canvas

NOTE: Your submission for this assignment should be a single **.cpp** file and a single **.pdf** file. The following naming convention should be used for naming your files: **firstname_lastname_165_assign8.cpp** and **firstname_lastname_165_assign8.pdf**. The pdf file that you submit should contain the screenshots of your sample runs of the program (see below). For example, if your first name is “James” and your last name is “Smith”, then your files should be named **James_Smith_165_assign8.cpp** and **James_Smith_165_assign8.pdf**.

COMMENTS (7.5% of programming assignment grade): Your program should have at least **ten (10)** different detailed comments explaining the different parts of your program. Each individual comment should be, at a minimum, a sentence explaining a particular part of your code. You should make each comment as

detailed as necessary to fully explain your code. You should also number each of your comments (i.e., comment 1, comment 2, etc.). **NOTE: My comments in the program do NOT count towards your ten comments!**

SAMPLE RUNS (7.5% of programming assignment grade): You should submit screenshots of at least **five (5)** different sample runs of your program. Each sample run needs to use a different array, and your sample runs should **NOT** be the same as the sample runs that are used in this write-up for the assignment. You should also number each of your sample runs (i.e., sample run 1, sample run 2, etc.). Each of your sample runs should be similar to this format:

```
// Constant for the array size  
const int SIZE = 11;
```

```
// An array of test values  
int test[SIZE] = {1, 2, 3, 3, 3, 2, 2, 1, 3, 4, 5};
```

 G:\DVC COMSC materials\Gaddis 8th ed materials\Programming Solutions\ISM\Chapter 09\spc9-8.exe

```
The mode of the test set is: 3
```

Each of your five (5) sample runs should contain two screenshots: 1) one screenshot of your code, showing the array that was used to run the program and 2) one screenshot of the corresponding output on the console screen.

You will be implementing the following function:

<p style="text-align: center;">getMode</p> <p>inputs: a Pointer to an integer array, and the size of the array</p> <p>Returns: the mode (as an integer)</p>
<pre>//***** // Function getMode * // This function returns the mode of the array pointed to * // by array. If there is no mode, the function returns -1. * //*****</pre>

NOTE: The mode is the number that occurs most frequently in a sequence. For example, in the sequence 1, 1, 2, 2, 2, 2, 3, 4, 4, 5, 5, 5 the mode is 2, because 2 occurs the most frequently. If none of the values occur more than once, then there is no mode, and the function should return -1.

If there is a tie for the mode, then the number that comes first is considered the mode. For example, in the set 1, 1, 2, 2, 3, 3, 4, 5, 6 there is a 3-way tie for the mode between 1, 2, and 3. In mathematics it is possible to have multiple modes, but for the purposes of this program there is only one mode at most.

The getMode function also calls the makeArray function, which creates and

returns a **dynamically allocated array** from the heap. This array stores the frequencies of the values from the test array in main. For example, if test[0]=1, then frequencies[0] will store **how many times** the value 1

occurs in the test array. If test[1]=3, then frequencies[1] will store how many times the value 3 occurs in the test array, and so on. In this sense, **the test array and the frequencies arrays are parallel arrays** -- the two arrays store related data, and you need to use the two arrays **together** in the getMode function to determine the mode of the test array.

IMPORTANT: Pointer notation should ALWAYS be used instead of subscript notation when accessing array elements. For example, when accessing element i in the test array, you should use *(test+i) instead of test[i]. Same idea for the frequencies array – you should use *(frequencies+i) instead of frequencies[i].

You will also be using the following function in your program, **but it has already been completed for you:**

makeArray
Input: the size of the array (as an integer) Returns: a Pointer to the new array that has been created
<pre>//***** // Function makeArray * // This function dynamically* allocates an array of ints * // and returns a pointer to it. The size parameter is * // the number of elements to allocate. * //*****</pre>

*The program should dynamically allocate memory from the heap for the array.

Sample Run 1:


```
// Constant for the array size  
const int SIZE = 11;  
  
// An array of test values  
int test[SIZE] = {1, 2, 3, 3, 3, 2, 2, 1, 3, 4, 5};
```

 G:\DVC COMSC materials\Gaddis 8th ed materials\Programming Solutions\ISM\Chapter 09\spc9-8.exe

The mode of the test set is: 3

Sample Run 2:


```
// Constant for the array size  
const int SIZE = 5;  
  
// An array of test values  
int test[SIZE] = {1, 2, 3, 4, 5};
```

 G:\DVC COMSC materials\Gaddis 8th ed materials\Programming Solutions\ISM\Chapter 09\spc9-8.exe

The test set has no mode.

Sample Run 3:

```
// Constant for the array size  
const int SIZE = 9;  
  
// An array of test values  
int test[SIZE] = {1, 1, 2, 2, 3, 3, 4, 5, 6};
```

 G:\DVC COMSC materials\Gaddis 8th ed materials\Programming Solutions\ISM\Chapter 09\spc9-8.exe

The mode of the test set is: 1