National University of Computer and Emerging Sciences, Lahore Campus

Course:	Object Oriented Programming	Course Code:	CS-1004
Program:	BS (Software Engineering)	Semester:	Spring 2023
Deadline:	6 Feb 22 (11:59 PM)	Total Marks:	10
Section:	BSE-2C		
	Homework-1		

Instruction/Notes:

1: Late submissions, even after a fraction of second will not be accepted.

2: Any kind of plagiarism will result into zero (whether you are provider or receiver)

Q: You need to provide the complete implementation as per of the given instructions. If there is any restriction in any of the function then you are supposed to provide the implementation without any violation of those restrictions, like if you asked to get the input only by de-referencing the pointer then you can't use any extra variable to get the input from user. There are few questions. You need to provide the answer in the comments of your code.

1: Declare the following in the main function:

- a pointer to integer (for dynamic array) initialized with NULL.
- an integer variable (for size) initialized with 0.

2: call the function **allocateMemory** from main function and pass both the parameters i.e., (pointer and integer variable) by reference

2.1: return type of this function should be void

2.2: it should receive by-reference pointer to integer and a by-reference integer in the parameters list

2.3: it should prompt the user to enter +ve size for the array until valid input is provided by the user and get the input in the variable received by reference.

2.3: allocate the memory using the pointer received **by-reference** in the parameters list

Q: What if both the parameters were by value?

3: create another pointer in main function and assign the address of last index location of the array. call the function **getData** from main function and pass the pointer 'p' and 'q' as parameters.

3.1: return type of this function should also be void

3.2: this function should receive both the parameters as **by-reference const pointers to integer**.

3.3: this function should be used to get input from user and it should only accept +ve integers treat -ve values as invalid data and prompt the user to provide valid input repeatedly for that instance.

3.4: You can't use array notation [] to get the data from user. Similarly, you only have two pointers pointing to first and last index locations. **You can't declare any extra variable to get the input from user.** You need to de-reference the pointer to store the data and keep in mind that both the pointers are constant.

Q: What if we receive the parameters as by reference const pointer to const int?

4: call the function growArray from main function and pass the pointer pointing to array and size in the parameters list

4.1: return type of this function should be void

4.2: This function should be used to grow the size of array. It should receive the parameters as by-reference pointer to integer and by-reference integer.

4.3: declare another temporary variable to get the new size of the array and prompt the user to enter the new size greater than previous size. If the provided data by user is smaller than previous size, then It should prompt the user repeatedly for valid data (valid data: Any value greater than the double of previous size)

4.4: Once you get the new_size, declare another pointer to integer say 'temp' in this function and allocate the memory equal to the amount of **new_size** using this 'temp' pointer.

4.5: Now copy the data from previous array (where the pointer in the parameters list is pointing) to this newly created array. (You are not allowed to use array notation i.e., [ ])

4.6: Since the size of newly created array is greater than previous so there are unfilled locations in the array even after copying the data from previous array. So get the data (accept only +ve values) for those unfilled locations from the user and again you can't declare any extra variable to get the input from user so de-referencing the pointer is the only available option.

4.7: once all the locations get filled now you need to de-allocate the memory where the pointer received in the parameters list is pointing and relocate the pointer to this newly create memory. Don't forget to update the by reference integer variable size received in parameters list.

(Q: Why de-allocation and relocation?)

(Q: What if any of the parameters i.e., pointer or the size variable or both of them were by-value).

5: call the function **printArray** from main function and pass the pointer pointing to the newly created array and the updated size value in the parameters list:

5.1: return type should be void

5.2: the parameters of this function should be by-reference const pointer to const integer and by-reference const int size.

5.3: print the data of the array. You can't use array notation to print the data.

Q: Why we are receiving the parameters by-reference?

Q: Why we are receiving by-reference const pointer?

Q: Why we are receiving by-reference const pointer to const integer?

Q: Why we are receiving by-reference const integer size?

6: call the function **shrinkArray** from main function and pass the pointer pointing to array and the size in the parameters list:

6.1: parameters of this function should be pointer to integer and by-reference int variable. since pointer is not received by reference so you need to decide the return type of this function.

6.2: this function should be used to shrink the array. So, declare a temporary int variable to get the new size from user. It should prompt the user repeatedly to enter the size until it gets valid input (valid input: Any value less than the half of current size)

6.3: create a pointer to integer say 'temp' and allocate the memory equal to the amount of new_size using this pointer. Copy the data from the array received in the parameters list into this newly created array. Make sure it should only store the values equal to the amount of new_size.

//Task: You need to make sure that the pointer in main function should also point to this newly create shrinked memory and there should not be any memory leak. Also, the size variable of main should also be updated.

//call the printArray function from main function to check whether you have successfully shrinked the memory or there is any issue?

7: call the function **getDistinct** from main function. This function should be used to return the array with only distinct elements. (Distinct elements = unique elements i.e., remove all the duplicate values). You need to pass the pointer pointing to the array and the size variable.

7.1: This function should receive two by-reference parameters i.e., by-reference pointer to integer and by reference integer size. Return type should be void.

7.2: You can use array notation [] in this function. First you need to count the distinct elements in the array and create a dynamic array by equal to the size of distinct elements.

7.3: Now copy only distinct elements from the previous array into this newly created array.

7.4: once all the distinct elements are copied then de-allocate the memory where the pointer in the parameters list is pointing i.e., (previous array) and relocate the pointer received in parameters list to this newly created memory. Don't forget to update the by reference integer "size" received in the parameters list. That size should be equal to the count of distinct elements.

8: call the **printArray** function from main function and pass the pointer pointing to the array of distinct elements and the size of the array. Description of print function is already provided.

9: In the end you need to call the function **releaseResources.** This function should be used to release the occupied resources i.e., dynamically allocated memory. The function should receive by-reference pointer to integer and de-allocate the memory. You need to make sure that there should not be any memory leak or the issue of dangling pointer after this function call.

Q: What if we receive by-reference const pointer to integer in the parameters list? Will there be any issue in de-allocation? Will there be any issue in relocating the pointer to NULL?