CMPT 130: Lab Work Week 7

Most of the questions below will ask you to write a function. It is your responsibility to create a suitable test main program for each function and test each of your function for correctness. In your test main program, create suitable array or string, manually analyze it (them) and think what your function should return; then run your program and see if your function returns correct answer.

In some of the questions, you may be asked to write a function that may require complicated loops or logical thinking. In such cases, it is advised you break the question down to some simple and high level steps and then implement the steps as functions; so that the main program may call a function and then that function may call another function to make problem solving easier. See the following complete example and use this idea as a blue print whenever you face problems seem too complicated to be solved in one function.

Example Question: Write a C++ program that reads two integers and prints all the prime numbers between the two integers inclusive. Assume the user input numbers are positive integers.

Solution: Design the main program to read two integers and re-arrange them so that you know which one is smaller and which one is larger integer. Then call a function to print all the prime numbers between them. When the function is designed, it should only loop from the smaller to the larger integer and for each integer in the loop, print if it is prime number as determined by yet another function that takes an integer and returns true or false depending if the integer number is prime or not.

```
bool isPrime(const int x)
{
     //Given integer x, if any number between 2 and x-1
     //divides x then x is not a prime.
     //If no number betwen 2 and x-1 divides x then x is prime
     for (int i = 2; i < x; i++)
         if (x % i == 0)
             return false;
     return true;
}
void printPrimes(const int a, const int b)
{
     //Loop between a and b; and for each number check if it
     //is prime by calling another function that tests for prime
     for (int k = a; k <= b; k++)
         if (isPrime(k) == true)
             cout << k << endl;
}
int main()
{
     //Step 1. Read two integers
     int a, b;
     cout << "Enter two positive integers ";
     cin >> a >> b;
     //Step 2. Re-arrange them so that a is the smaller and b the larger
     if (a > b)
     {
         int temp = a;
         a = b;
         b = temp;
     }
     //Step 3. Print all the prime numbers between them
     cout << "All the prime numbers between the numbers inclusive are..." << endl;
     printPrimes(a, b);

     system("Pause");
     return 0;
}
```

Counting Problems

1. Write a program that creates a C++ static array of integers of length 10, populates the elements of the array with random integers and then prints how many of them are even integers and how many are odd.

2. Write a program that creates a C++ static array of integers of length 10, populates the elements of the array with random integers and then prints how many of them are prime integers and how many are not prime integers.

3. Write a program that creates a C++ static array of floats of length 10, populates the elements of the array with random floats in the range -1.0 and 1.0 and then prints how many of them are positive floats and how many are negative floats.

4. Write a function named **evenCounter** that takes an array of integers and its size as arguments and returns the number of even elements in the array.

5. Write a function named **primeCounter** that takes an array of integers and its size as arguments and returns the number of elements of the array that are prime numbers. Assume the array contains positive integers greater than 1.

6. Write a function named **vowelCounter** that takes an array of characters and its size and returns the number of vowel characters in the array. Vowels are 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O' and 'U'

7. Write a C++ function named **countElements** that takes three arguments: an array of integers, its size, and an integer and that returns the number of elements in the array that are equal to the integer argument.

8. Write a C++ function named **countElementsFromIndex** that takes four arguments: an array of integers, its size, an integer value x, and an integer index. Your function must search the integer value x in the array starting from the index argument upto the last element of the array and return the number of elements in the array that are equal to the integer argument. For example give the following code

```
const int size = 11;
int A[]= {5, 7, 8, 0, 5, 4, 1, 5, 6, 9, 5};
int searchValue = 5;
int startIndex = 2;
```

   Then the function call **countElementsFromIndex(A, size, searchValue, startIndex);** must return 3 because there are 3 elements equal to 5 starting from index 2. Of course in total there are 4 elements equal to 5 but because the search starts at the index given by the index argument, then the function will find only 3 elements equal to 5.

9. Write a C++ function named **countCommonElements** that takes four arguments: an array of integers, its size, a second array of integers, and its size. Your function must return the number of elements of the first

array that are also found in the second array. Assume each of the arrays contains distinct (i.e. different) elements.

10. [Challenge] Write a function named **distinctElementsArray** that takes an array of integers and its size as arguments and returns true if the array contains distinct (i.e. different) elements; otherwise returns false.

Squeezing Problems

11. Write a program that creates a C++ static array of integers of length 10, populates the elements of the array with random integers in the range [-10, 10] and then prints the sum of all the elements.

12. Write a program that creates a C++ static array of integers of length 10, populates the elements of the array with random integers in the range [-10, 10] and then prints the sum of the absolute values of all the elements.

13. Write a program that creates a C++ static array of integers of length 10, populates the elements of the array with random integers in the range [-10, 10] and then prints the absolute value of the sum of all the elements.

14. Write a program that creates a C++ static array of integers of length 10, populates the elements of the array with random integers in the range [-10, 10] and then prints the product of all the elements.

15. Write a program that creates a C++ static array of integers of length 10, populates the elements of the array with random integers in the range [-10, 10] and then prints the product of the absolute values of all the elements.

16. Write a program that creates a C++ static array of integers of length 10, populates the elements of the array with random integers in the range [-10, 10] and then prints the absolute value of the product of all the elements.

17. Write a program that creates a C++ static array of integers of length 10, populates the elements of the array with random integers in the range [-10, 10] and then prints the maximum and minimum elements of the array.

18. Write a C++ function named **maxElement** that takes an array of integers and its size as arguments and returns the maximum element.

19. Write a C++ function named **minElement** that takes an array of integers and its size as arguments and returns the minimum element.

20. Write a C++ function named **sumArray** that takes an array of floats and its size as arguments and returns the sum of the elements in the array.

21. Write a C++ function named **productArray** that takes an array of floats and its size as arguments and returns the product of the elements in the array.

Existence of specific element Problems

22. Write a C++ function named **isFound** that takes three arguments: an array of integers, its size, and an integer and that returns true if the integer argument is found in the array; otherwise return false.

23. Write a C++ function named **isFoundFromIndex** that takes four arguments: an array of integers, its size, an integer x, and an integer index. Your function must return true if the x is found inside the array starting from the index argument up to the last element of the array; otherwise return false.

24. Write a function named **containsEven** that takes an array of integers and its size; and that returns true if the array contains at least one even number element otherwise return false.

25. Write a function named **containsDigit** that takes an array of characters and its size; and that returns true if the array contains a digit otherwise return false.

26. Write a function named **containsVowel** that takes an array of characters and its size; and that returns true if the array contains a vowel otherwise return false.

27. Write a function named **containsLowerCase** that takes an array of characters and its size; and that returns true if the array contains a lower case English letter otherwise return false.

28. Write a function named **containsUpperCase** that takes an array of characters and its size; and that returns true if the array contains an upper case English letter otherwise return false.

29. Write a function named **isAlpha** that takes an array of characters and its size; and that returns true if the array contains only English alphabets; otherwise return false.

30. Write a function named **isAlphaNumeric** that takes an array of characters and its size; and that returns true if the array contains only English alphabets and digits; otherwise return false.

31. Write a function named **containsPrime** that takes an array of integers and its size; and that returns true if the array contains at least one prime number element otherwise return false.

32. Write a function named **containsComposite** that takes an array of integers and its size; and that returns true if the array contains at least one composite number element otherwise return false.

33. Write a program that creates a C++ static array of integers of length 10, populates the elements of the array with random integers in the range [-10, 10] and then prints the message **"a negative number found in the array"** if the array contains at least one negative integer and prints the message **"No negative number found in the array"** if the array does not contain any negative number.

34. Write a program that creates a C++ static array of integers of length 10, populates the elements of the array with random integers and then prints the message **"a prime number found in the array"** if the array contains at least one prime number and prints the message **"No prime number found in the array"** if the array does not contain any prime number.

35. Write a program that creates a C++ static array of characters of length 10, populates the elements of the array with random characters and then prints the message **"a digit found in the array"** if the array contains at least one digit and prints the message **"No digit found in the array"** if the array does not contain any digit. For random characters consider only those characters whose ASCII code is in the range [33, 126] in decimal.

36. Write a program that creates a C++ static array of characters of length 10, populates the elements of the array with random characters and then prints the message **"an alphabet found in the array"** if the array contains at least one alphabet and prints the message **"No alphabet found in the array"** if the array does not contain any alphabet. For random characters consider only those characters whose ASCII code is in the range [33, 126] in decimal.

37. Write a function named **isIncreasing** that takes an array of integers and its size and that returns true if the elements of the array are in increasing order; otherwise return false.

C++ Strings

38. Write a function named **randomName** that takes no argument and returns a random name string of 10 characters made up of only English alphabets. The first character of the name must be uppercase while all the remaining characters must be lowercase.

39. Write a C++ function named **isFound** that takes two arguments: a string and a character. Your function must return true if the character argument is found in the string; otherwise it must return false.

40. Write a function named **vowelCounter** that takes a C++ string and returns the number of vowel characters in the C++ string.

41. Write a C++ function named **countCharacter** that takes a string and a character as arguments and returns the number of times the character is found in the string.

42. Write a C++ function named isDistinct that takes a string argument and returns true if the string argument contains distinct characters otherwise returns false.

43. Write a C++ function named **countCharacterFromIndex** that takes a string, a character and an integer index as arguments and returns the number of times the character is found in the string starting from the given index argument.

44. Write a C++ function named **countCommonChars** that takes two string arguments s1 and s2 and then returns the number of characters of s1 found in s2. For simplicity you can assume the string s1 contains distinct characters and also the string s2 contains distinct characters. Although strictly speaking you don't need this assumption. So if it makes the problem easy for you, then assume so.

45. Write a C++ function named **commonCharsString** that takes two string arguments and returns a new string made up of all the characters of s1 that are found in s2. For simplicity you can assume the string s1 contains distinct characters and also the string s2 contains distinct characters. Although strictly speaking you don't need this assumption. So if it makes the problem easy for you, then assume so.

46. Write a program that declares an array of 5 string data type, initializes each element of the array with a random name by calling your function **randomName()** defined in Q38, and then prints the name that comes first in the order of the string elements and the name that comes last in the order of the string elements.

47. Write a function named **increasingOrderStrings** that takes an array of string data type and its size and then returns true if the elements are in increasing order otherwise returns false.