CMPT 125: Week 3 Lab Work

C++ static arrays, cstrings, C++ strings, and algorithms

Most of the questions below will ask you to define a function. It is your responsibility to create a suitable test main program for each function and test your functions for correctness. In your test main program, create suitable array or string, manually analyze it and think what your function should perform or return. Then run your program and see if your function does its job correctly.

In some of the questions, you may be asked to define a function that may require complicated loops or logical thinking. In such cases, it is advised you break down your solution 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 other supporting/helper function(s) to make the problem solving easier. See the following complete example and use this idea as a blue print whenever you face problems that seem too complicated to be solved in one function.

Example Question: Define a function that takes an array of integers and prints the elements of the array that are prime numbers. Assume the array is populated with random integers in the range [2, 100].

Solution: Design the test main program to create an array of integers of any size you wish and populate it with random integers in the range [2, 100]. Then call the function to print the elements of the array that are prime numbers. This function should be designed to loop on the elements of the array and for each element of the array print it only 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 a 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 arr[], const int size)
{
     //Loop on the elements of the array and for each element check if it
     //is prime by calling another function that tests for prime.
     //If it is prime then print it. Otherwise don't print it.
     for (int k = 0; k < size; k++)
         if (isPrime(arr[k]) == true)
             cout << arr[k] << endl;
}
int main()
{
     //Step 1. Create an array
     int A[10];
     //Step 2. Populate the array with random integers in the range 2 to 100
     for (int i = 0; i < 10; i++)
         A[i] = rand() % 99 + 2;
     //Step 3. Print the elements of the array
     cout << "The elements of the array are..." << endl;
     for (int i = 0; i < 10; i++)
         cout << A[i] << endl;
     //Step 4. Print the elements of the array that are prime numbers
     cout << "The elements of the array that are prime numbers are..." << endl;
     printPrimes(A, 10);

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

1. Write a C++ program that declares a static array of int data type of size 10, populates the elements of the array with random integers in the range [-25, 25], prints the elements of the array, reverses the array, and finally prints the elements of the reversed array.
2. Write a program that declares a static array of char data type of size 11, populates the first ten elements of the array with random charscters in the range ['a', 'z'] and assigns the last element of the array a NULL character, prints the cstring, reverses the cstring, and finally prints the reversed cstring.
3. Write a program that declares a C++ string, concatenates ten random characters in the range ['a', 'z'] to the C++ string, prints the C++ string, reverses the C++string, and finally prints the reversed C++ string.
4. Write a program that declares a C++ string, concatenates ten random characters in the range ['a', 'z'] to the C++ string, prints the C++ string, removes the vowel characters from the string, and finally prints the modified C++ string.
5. Write a complete 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.
6. Define a C++ function named **isFound** that takes three arguments: an array of integers, its size, and an integer value and that returns true if the integer argument is found in the array; otherwise return false.
7. Define a C++ function named **isFound** that takes two arguments: a cstring and a character. Your function must return true if the character argument is found in the cstring; otherwise it must return false. Hint: use the cstringlen function defined in the lecture when you answer this question.
8. Define a C++ function named **isFound** that takes two arguments: a C++ string and a character. Your function must return true if the character argument is found in the string; otherwise it must return false.
9. Define a function named **vowelCounter** that takes a cstring and returns the number of vowel characters in the cstring. Vowels are 'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O' and 'U'
10. Define a C++ function named
11. that takes a C++ string and a character as arguments and returns the number of times the character is found in the string.
12. Define a function named **randomName_string** that takes no argument and returns a C++ string of ten characters made up of only English alphabets. The first character of the name must be upper case while all the remaining characters must be lower case.
13. Define a function named **containsDigit** that takes a cstring and that returns true if the cstring contains at least one digit character otherwise return false.
14. Define a function named **allDigit** that takes a cstring and that returns true if all the characters of the cstring are digit characters; otherwise return false.
15. Define 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.
16. Define a function named **allPrime** that takes an array of integers and its size; and that returns true if all the elements of the array are prime numbers otherwise return false.

17. Write a complete program that creates a cstring containing ten random printable characters whose ASCII codes are in the range [33, 122], prints the random cstring, and finally prints the message **"an alphabet found in the cstring"** if the cstring contains at least one alphabet character and prints the message **"No alphabet found in the cstring"** if the cstring does not contain any alphabet.

18. Define 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. What is the pre-condition for the function?

19. Define a C++ function named **isDistinct** that takes a cstring argument and returns true if the cstring argument contains distinct characters otherwise returns false. What is the pre-condition for the function?

20. Define a C++ function named **isDistinct** that takes a C++ string argument and returns true if the C++ string argument contains distinct characters otherwise returns false. What is the pre-condition for the function?

21. Define 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. What is the pre-condition for the function?

22. Define a function named **isIncreasing** that takes a cstring and that returns true if the printable characters of the cstring are in increasing order; otherwise return false. What is the pre-condition for the function?

23. Define a function named **isIncreasing** that takes a C++ string and that returns true if the elements of the string are in increasing order; otherwise return false. What is the pre-condition for the function?

24. Define a function named **isIncreasing** that takes an array of C++ string data type and its size and then returns true if the elements are in increasing order otherwise returns false. What is the pre-condition for the function?

25. Define a function named **reArrange** that takes an array of integers and its size as arguments and that rearranges the array such that the even integer elements of the array are placed at the beginning of the array and the odd integer elements of the array are placed at the end of the array. A sample test program and its output is given below to help you understand the question better.

```
int main()
{
	//Define an array
	const int SIZE = 20;
	int A[SIZE];

	//Populate the array with some integers in the range [0, 29]
	for (int i = 0; i < SIZE; i++)
		A[i] = rand() % 30;

	//Print the elements of the array
	cout << "Originally the elements of the array are" << endl;
	for (int i = 0; i < SIZE; i++)
		cout << A[i] << "   ";
	cout << endl;

	//Re-arrange the array so that even integers are at the beginning
	reArrangeArray(A, SIZE);

	//Print the elements of the array
	cout << "After re-arranging the elements of the array are" << endl;
	for (int i = 0; i < SIZE; i++)
```

```
            cout << A[i] << "   ";
        cout << endl;

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

OUTPUT

```
Originally the elements of the array are
11 17 4 10 29 4 18 18 22 14 5 5 1 27 1 11 25 2 27 6
After re-arranging the elements of the array are
4 10 4 18 18 22 14 2 6 17 5 5 1 27 1 11 25 29 27 11
Press any key to continue . . .
```

Please note that the order of the elements within the same group is not important.

26. Define a C++ function named **commonCharsString** that takes two string arguments s1 and s2 and returns a new C++ 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.

27. Define a C++ function named **`findIndex`** that takes a static array of integers A and its size as arguments with the condition that there exists an index **k** in the array such that the all elements A[0], A[1],...,A[k] are all even integer numbers and all the remaining elements are all odd integer numbers. Your function must return the index **k**. You must use a logarithmic algorithm. Assume the size of the array is greater than 2.

Here is a test program to help you test your function.

```
int main()
{
        int A[10] = {2, 8, 16, 4, 14, 8, 10, 6, 5, 9};
        int k = findIndex(A, 10);
        cout << "The last even element is found at index " << k << endl;
        system("Pause");
        return 0;
}
```

28. Given two arrays **A** and **B** with equal size **n**, define a C++ function that returns true if every element of A is found in B; otherwise returns false.

In your function there will be an **if-statement** that compares an element in A with an element in B, how many times does, in the worst case, that **if-statement** get executed? Give your answer in terms of **n**. Give the complexity of your algorithm in Big-O notation. Classify your algorithm as linear, logarithmic, quadratic, or some other class of functions. Assume the arrays **A** and **B** are not sorted.

29. Repeat question 27 above but this time assume the array **A** is not sorted but the array **B** is sorted.

30. Repeat question 27 above but this time assume the array **A** is sorted but the array **B** is not sorted.

31. Repeat question 27 above but this time assume both the arrays **A** and **B** are sorted.