FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES AND MULTIMEDIA ARTS

CCS0015L
(DATA STRUCTURES AND ALGORITHMS)

EXERCISE
6.2

RECURSION

Student Name / Group Name:	Gerard Eric B. Doroja
Members (if Group):	**Name** / **Role**
Section:	BSITBA – TB02
Professor:	Mr. Ronel Ramos

I. PROGRAM OUTCOME/S (PO) ADDRESSED BY THE LABORATORY EXERCISE

- Identify, analyze and solve computing problems using fundamental principles of mathematics and computing sciences. [PO: B]

II. COURSE LEARNING OUTCOME/S (CLO) ADDRESSED BY THE LABORATORY EXERCISE

- Apply the fundamental principles of data structures and algorithms: concepts of abstract data; types of common data structures used; description, properties, and storage allocation of data structures. [CLO: 2]

III. INTENDED LEARNING OUTCOME/S (ILO) OF THE LABORATORY EXERCISE

At the end of this exercise, students must be able to:

- Learn how to create a base case and inductive step.
- Implement recursion in solving programming problems that require it.

IV. BACKGROUND INFORMATION

What is Recursion?

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily. Examples of such problems are Towers of Hanoi (TOH), In-order/Preorder/Post-order Tree Traversals, DFS of Graph, etc.

What is base condition in recursion?

In recursive program, the solution to base case is provided and solution of bigger problem is expressed in terms of smaller problems.

```
int fact(int n)
{
    if (n < = 1) // base case
        return 1;
    else
        return n*fact(n-1);
}
```

In the above example, base case for n < = 1 is defined and larger value of number can be solved by converting to smaller one till base case is reached.

Need of Recursion

Recursion is an amazing technique with the help of which we can reduce the length of our code and make it easier to read and write. It has certain advantages over the iteration technique which will be discussed later. A task that can be defined with its similar subtask, recursion is one of the best solutions for it. For example, The Factorial of a number.

V. LABORATORY ACTIVITY

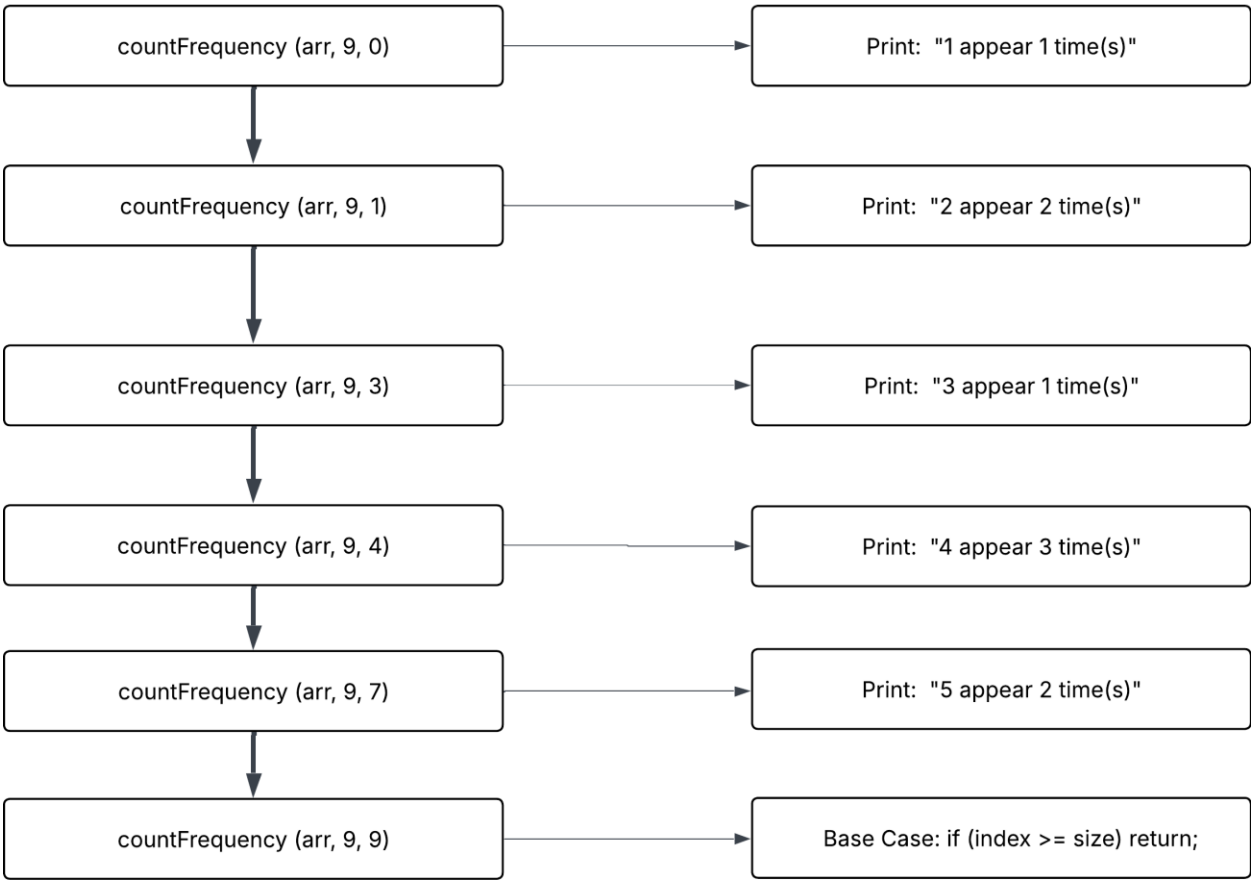
ACTIVITY 6.2: Duplicates

After writing the C++ program that finds the frequency of each element in a sorted array containing duplicates using recursion, create a diagram that demonstrates the recursion.

Sample Output:
If the array given is: int arr[] = {1, 2, 2, 3, 4, 4, 4, 5, 5};

```
1 appears 1 time(s)
2 appears 2 time(s)
3 appears 1 time(s)
4 appears 3 time(s)
5 appears 2 time(s)
```

If the array given is: int arr[] = {1, 2, 2, 3, 4, 4, 4, 5, 5};



VI. QUESTION AND ANSWER

Directions: Briefly answer the questions below.

- Explain the advantages and disadvantages of using recursion.

Recursion is a very useful program design method whereby a function can be called upon itself in an effort to solve the problem that is to be solved by reducing the problem to small manageable prayer problems. A key difference where recursion can prove helpful is that, unlike iteration, recursion may make the solution to a complex problem such as a tree traversal, graph search, or a mathematical computation (calculate a factorial or the Fibonacci sequence) much more intuitive and its reading easier. An advantage of recursive solutions is that they may keep the natural structure of the problem and thus be easier to understand and design. Nevertheless, recursion has its own drawbacks as well. Every call to the recursive function will add another frame to the call stack, which might easily fill up the memory and results in the stack overflow errors when the recursion is too deep or in case the base case has not been reached. Recursive functions may also be more computationally costly than their iterative equivalents because of the over-head of calling functions recursively and the danger of recalculating work, especially in case the problem was not optimized through methods such as memoization. Also, debugging is a little more tricky and not so ideal at least with beginners, simply because it is not easy to flow through the execution of a recursive piece of code as compared to simple loops.

https://www.simplilearn.com/tutorials/cpp-tutorial/what-is-recursion-in-cpp
https://www.codecademy.com/resources/blog/recursion/

- Discuss how base case is identified in a given problem when implementing recursion.

Deriving the base case should be one of the key elements in getting the correct implementation of recursion since this is what stops the recursive process. Base case is usually the easiest example of the problem: the one which can be solved immediately without any additional recursions. What you need to do to find the base case is study the problem so that you understand what is the single smallest input or what is the simplest condition where the answer has been provided or which you can calculate instantly. As an example, when using a recursion method, to calculate the factorial of a number, the base case is where the number is either 0 or 1, as the value of the factorial of both is 1. In a recursive search through a linked list, the base case might be reaching the end of the list (a null pointer) or finding the target value. It is crucial that each recursive call progresses toward the base case; otherwise, the recursion could continue indefinitely, leading to errors and inefficient code. Clearly defining and correctly implementing the base case ensures that the recursive function will terminate properly and produce the correct result.

https://www.simplilearn.com/tutorials/cpp-tutorial/what-is-recursion-in-cpp
https://cbselibrary.com/advantages-and-disadvantages-of-recursion/

VII. REFERENCES

- Wittenberg, Lee.(2018). Data structures and algorithms in C++. s.l.: Mercury Learning
- Baka, Benjamin(2017). Python data structures and algorithms : improve the performance and speed of your applications. Birmingham, U.K : Packt Publishing
- Downey, Allen.(2017). Think data structures : algorithms and information retrieval in Java. Sebastopol, CA: O'Reilly
- Chang, Kung-Hua(2017). Data Structures Practice Problems for C++ Beginners. S.l : Simple and Example
- Hemant Jain(2017). Problem Solving in Data Structures & Algorithms Using C++: Programming Interview Guide. USA: CreateSpace Independent Publishing Platform

RUBRIC:

Criteria	4	3	2	1	Score
Solution(x5)	A completed solution runs without errors. It meets all the specifications and works for all test data.	A completed solution is tested and runs but does not meet all the specifications nd/or work for all test data.	A completed solution is implemented on the required platform, and uses the compiler specified. It runs, but has logical errors.	An incomplete solution is implemented on the required platform. It does not compile and/or run.	
Program Design(x3)	The program design uses appropriate structures. The overall program design is appropriate.	The program design generally uses appropriate structures. Program elements exhibit good design.	Not all of the selected structures are appropriate. Some of the program elements are appropriately designed.	Few of the selected structures are appropriate. Program elements are not well designed.	
Completeness of Document(x2)	All required parts of the document are complete and correct(code, output of screenshots)	All required parts in the document are present and correct but not complete.	There are few parts of the document are missing but the rest are complete and correct.	Most of the parts of the document are missing and incorrect.	

					Total