CS4402 Discussion 4

**Question 1:**We learned that when calling subprograms or functions that we can pass data to the subprogram or function by ‘value’ or by ‘reference’.   Describe what each approach is, how it works and what the potential security disadvantage is when passing parameters by reference.

You are expected to make a minimum of 3 responses to your fellow students’ posts.

**Question 2:** Recursion is one way of implementing recursion in programs to solve complex problems.  Select one of the following three common computer science problems and describe how recursion can be used to solve this problem more efficiently (sorting, minimum cost spanning tree, knapsack problem).  You must generally describe the algorithm that would be used to solve the problem and detail how recursion makes the algorithm more asymptotically efficient.

You are expected to make a minimum of 3 responses to your fellow students’ posts.

**Question 1**

According to (Ben-Ari, 2006), there are two approaches to passing the data into the subprogram.

The first one is pass-by-value, this can be categorized as the copy-in and copy-out semantics.

The value is direct get involved with the calculation. The value of the actual parameter is copied to the allocated memory location of the formal parameter that is declared in the subprogram.

The copy-in is the safest mechanism for parameter passing. The only value is bringing over for the subprogram processing. The actual parameters can be constants, variables, or expressions.

But there are occasionally needs that we need to modify the actual parameter within the subprogram.

1. The reason for our need that call-by reference is below:
2. A function can only return one result but we might need to archive more changes during the process.
3. The subprogram might just need to modify data instead of return a value.
4. The actual parameter might be large and inefficient to copy.

Thus, the call-by-reference semantics is developed. This semantics will pass the address of the actual parameter and access the parameter directly. So, it resolves the large parameter issue.

There are potential security concerns that may arise due to the aliasing issue. That the same variable is known by more than one name. this is a disadvantage that makes the semantics inherently unsafe. The subprogram might accidently smear the actual parameter then it wrongly processes the value in the address as a different data type.

However, this can be improved by adapting the strong data type checks.

**Question 2**

The knapsack problem is a classic optimization problem where you have a set of items, each with a weight and a value, and you want to find the most valuable combination of items to fit into a knapsack (a limited-capacity bag) without exceeding its weight limit(*Knapsack Problem - Wikipedia*, n.d.).

Recursion can help solve this problem by breaking it down into smaller subproblems. Here's a high-level overview of how it works:

Base Case: You start with a base case. In the context of the knapsack problem, this could be something like having an empty knapsack or no items left to consider.

Recursive Step: Then, you define a recursive function. In this case, it's a function that calculates the maximum value that can be obtained with the current item set and the remaining capacity of the knapsack.

Combine Solutions: The recursive function combines solutions with subproblems. For each item, it has two choices: include it in the knapsack or leave it out. It calculates the value for both options and chooses the one with the maximum value.

Return the Result: Finally, you return the maximum value that can be obtained with the entire set of items and the knapsack's capacity.

The recursive approach explores all possible combinations of items, which makes it a bit inefficient for large problem sizes. However, it's a great way to understand the problem conceptually.

**Reference**

Ben-Ari, M. (2006). Understanding programming languages. *Weizmann Institute of Science., Rehovot, Israel*.

*Knapsack problem - Wikipedia*. (n.d.). Retrieved October 1, 2023, from https://en.wikipedia.org/wiki/Knapsack\_problem