Assignment 9: Stack Overflow, Anatomy Of Recursion

Assignment Description:

Dive into the inner workings of recursion, understand its limits, and explore the anatomy of stack overflow errors.

**Questions**:

Stack Overflow: Explain what a stack overflow error is in the context of recursive functions. What causes stack overflow errors, and how can they be prevented or mitigated?

**Solution** 1:

A stack overflow error occurs when the call stack, which keeps track of function calls, becomes too deep, causing it to exceed its memory limits. This is typically caused by infinite or excessively deep recursion. To prevent or mitigate stack overflow errors, it's essential to have a proper base case in recursive functions and limit the depth of recursion by optimizing code or using iterative solutions.

Question 2: Anatomy Of Recursion

Describe the internal workings of recursive function calls, including the call stack, function frames, and how data is stored during recursion. Provide a Java code example to illustrate this process.

**Solution** 2:

Recursion involves creating function frames on the call stack for each recursive call. Each frame stores local variables and the return address. The call stack follows the Last-In-First-Out (LIFO) principle. Here's a Java code example to illustrate recursion and the call stack:

java

public int factorial(int n) {

if (n <= 1) {

return 1;

} else {

int result = n \* factorial(n - 1);

return result;

}

}

Question 3: Exercise: Factorial

Implement a Java recursive function to calculate the factorial of a non-negative integer. Explain the base case and the recursive step in your solution.

**Solution** 3:

Here's a Java recursive function to calculate the factorial of a non-negative integer:

java

public int factorial(int n) {

if (n == 0) { // Base case

return 1;

} else {

// Recursive step: n! = n \* (n-1)!

return n \* factorial(n - 1);

}

}

The base case is when n equals 0, and the recursive step calculates the factorial by multiplying n with the factorial of n-1.