Assignment Solution: Recursion -2

1. **Write a program to calculate the sum of the digits of a given positive integer using recursion.**

Ans:

```cpp
#include <iostream>
using namespace std;

// Function to calculate the sum of digits using recursion
int sumOfDigits(int n) {
    if (n == 0) {
        return 0; // Base case: if n is 0, return 0
    }
    return (n % 10) + sumOfDigits(n / 10);
}

int main() {
    int n;
    cout << "Enter a positive integer: ";
    cin >> n;

    int sum = sumOfDigits(n);
    cout << "The sum of the digits of " << n << " is: " << sum << endl;

    return 0;
}
```

2. **Write a program to calculate the reverse of a given positive integer using recursion.**

Ans:

```cpp
#include <iostream>
#include <cmath>
using namespace std;
```

```
// Helper function to calculate the number of digits in n
int numberOfDigits(int n) {
    if (n == 0) {
        return 0;
    }
    return 1 + numberOfDigits(n / 10);
}

// Function to calculate the reverse of a number using recursion
int reverseNumber(int n, int digits) {
    if (n == 0) {
        return 0;
    }
    return (n % 10) * pow(10, digits - 1) + reverseNumber(n / 10, digits - 1);
}

int main() {
    int n;
    cout << "Enter a positive integer: ";
    cin >> n;

    int digits = numberOfDigits(n);
    int reversedNumber = reverseNumber(n, digits);

    cout << "The reverse of " << n << " is: " << reversedNumber << endl;

    return 0;
}
```

3. **Given an integer num, return the number of steps to reduce it to zero. In one step, if the current number is even, you have to divide it by 2, otherwise, you have to subtract 1 from it. [Leetcode 1342]**

Ans:

```
class Solution {
public:
    int numberOfSteps(int num) { // number of steps to reach 0
        if(num==0) return 0; // base case 1
        if(num==1) return 1; // base case 2
        return numberOfSteps(num/2)+(num%2==0?1:2); // recursive case and add 1 if num is
odd else add 2
    }
};
```

4. Predict the output:

```
int fun(int n) {
 if (n <= 1) return 1;
  if (n % 2 == 0) return fun(n / 2);
  return fun(n / 2) + fun(n / 2 + 1);
}
```

Ans:

For n = 0:
fun(0) returns 1 (since n <= 1).
For n = 1:
fun(1) returns 1 (since n <= 1).
For n = 2:
fun(2) returns fun(1) which is 1 (since n is even and fun(1) returns 1).
For n = 3:
fun(3) returns fun(1) + fun(2) which is 1 + 1 = 2 (since n is odd, fun(1) and fun(2) both return 1).
For n = 4:
fun(4) returns fun(2) which is 1 (since n is even and fun(2) returns 1).