Review Activity 6

Recursion in C/C++

1) Write a recursive function that takes a single integer as input and then prints that integer in reverse (e.g., for 12345 as input, the function prints 54321).

2) Implement a recursive function that takes a single integer as input and then prints the digits of that integer one digit per line (e.g., for 12345 as input, the function prints "1 \n 2 \n 3 \n 4 \n 5 \n" with no spaces).

3) Write a recursive function that will compute the sum of a series from 1 to n for a given n (i.e., 1 + 2 + … + n).

4) Implement a recursive function that will print out the series of squares from 1 to n for a given n (i.e., 1^2, 2^2 … n^2).

5) Write a recursive function that finds the largest integer divisor of a given integer other than itself (e.g., for 15 as input, the function returns 5).

6) Write a recursive function that checks if a given string is a palindrome (e.g., "aba" is a palindrome).

7) Write a recursive function that determines if the given number is a prime number. The smallest prime number is two. The function takes a single `unsigned int` as input, and returns `true` if the number is a prime and `false` otherwise. You need to use recursive helper functions.

Q1.

```
void printReverseInteger(int num){
    if(num %10 == 0)
        return;
    else if(num < 0){
        cout << "-";
        printReverseInteger(-1*num);
    }
    else{
        cout << num % 10;
        printReverseInteger(num/10);
    }
}
```

Q2.

```
void printDigitPerLine(int num){
    if(num < 10){
        cout << num << endl;
        return;
    }
    else if(num < 0){
        cout << "-" << endl;
        printDigitPerLine(-1*num);
    }
    else{
        printDigitPerLine(num/10);
        cout << num % 10<< endl;
    }
}
```

Q3.

```
int calculateSeries(unsigned int n){
    if( n <= 1)
        return 1;
    else
        return n + calculateSeries(n-1);
}
```

Q4

```
void squareSeries(unsigned int n){
    if(n<=1){
        cout << 1 << " ";
        return;
    }
    else{
        squareSeries(n-1);
        cout << pow(n,2) << " ";
    }
}
```

Q5

```
int findDivisor(int num,  int best_divisor){ // Pass in any number that is num-1
    if (best_divisor >= num || best_divisor < 1){
        return findDivisor( num , num-1);
    }
    if(num % best_divisor == 0)
        return best_divisor;
    else{
        return findDivisor(num, best_divisor -1);
    }
}
```

Q6.

```
bool Palindrome(string palindrome){
    if(palindrome.length() <= 1)
        return true;
    else{
        if(palindrome[0] == palindrome[palindrome.length() -1] )
            return Palindrome(palindrome.substr(1, palindrome.length() -2));
        return false;
    }
}
```

Q7

```
int findDivisor(int num,  int best_divisor){ // Pass in any number that is num-1
    if (best_divisor >= num || best_divisor < 1){
        return findDivisor( num , num-1);
    }
    if(num % best_divisor == 0)
        return best_divisor;
    else{
        return findDivisor(num, best_divisor -1);
    }
}

bool isPrimeNumber(int num){
    // return one if can not find divisor from num-1 to 2
    return (findDivisor(num, num -1) == 1);
}
```