Review Activity 6 Solutions

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).

```
	void print_in_reverse(int val) {
		if (val < 10) { // base case: handle single digits
			cout << val;
		} else { // recursive case: handle non-single digits
			cout << val % 10;
			print_in_reverse(val / 10);
		}
	}
```

```
	void print_in_reverse2(int val) {
		if (val < 0) {// optional: handle negative values
			cout << '-';
			print_in_reverse(-val);
		} else if (val < 10) { // base case: handle single digits
			cout << val;
		} else { // recursive case: handle non-single digits
			cout << val % 10;
			print_in_reverse(val / 10);
		}
	}
```

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).

```
	void print_digits(unsigned int val) { // only non-negative values allowed
		if (val < 10) { // base case: handle single digits
			cout << val  << endl;
		} else { // recursive case: handle non-single digits
			print_digits(val / 10);
			cout << val % 10 << endl;
		}
	}
```

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).

```
	int compute_series(unsigned int n) {
		if (n <= 1) { // base case: n <= 1
			return n;
		} else { // recursive case: n > 1
			return n + compute_series(n - 1);
		}
	}
```

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).

```
	void print_series(unsigned int n) {
		if (n == 0) { // base case: n = 0
			// do nothing
		} else if (n == 1) { // base case: n = 1
			cout << 1 << " ";
		} else { // recursive case: n != 1
			print_series(n - 1);
			cout << n * n << " ";
		}
	}
```

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).

```
	int find_divisor(unsigned int n, unsigned int d) {
		if (n <= 1) { // base case: n <= 1
			return n;
		} else if (n % d == 0) { // base case: n > 1 and n divisible by d
			return d;
		} else { // recursive case: n > 1 and n not divisible by d
			return find_divisor(n, d - 1);
		}
	}

	int find_divisor(unsigned int n) {
		return find_divisor(n, n / 2); // overloaded helper function
	}
```

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

```
	bool is_palindrome(string val) {
		if (val.length() <= 1) { // base case: length <= 1
			return true;
		} else { // recursive case: length > 1
			bool digits_equal = (val[0] == val[val.length() - 1]);
			return digits_equal && is_palindrome(val.substr(1, val.length() - 2));
		}
	}
```

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.

```
int find_divisor(unsigned int n, unsigned int d) {
	if (n <= 1) { // base case: n <= 1
		return n;
	} else if (n % d == 0) { // base case: n > 1 and n divisible by d
		return d;
	} else { // recursive case: n > 1 and n not divisible by d
		return find_divisor(n, d - 1);
	}
}

bool is_prime(unsigned int n) {
	// check if the number is greater than one and
	// if its largest divisor other than itself is one
	return (n > 1 && find_divisor(n, n/2) == 1);
}

// the following is not needed; provided for testing purposes
int main() {
	int test_val = 5;
	cout << test_val << (is_prime(test_val) ? " is a prime" : " is not a prime");

	return 0;
}
```