CSC 212: Data Structures and Abstractions

Fall 2018

University of Rhode Island

Weekly Problem Set #5

Due Thursday 10/11 at the beginning of class. Please turn in neat, and organized, answers hand-written on standard-sized paper **without any fringe**. At the top of each sheet you hand in, please write your name, and ID.

1. Write a recursive function that sums all of the elements of a given array, matching this signature:

```
int sum(int* arr, int n);
```

```
int sum(int* arr, int n) {
    if (n == 0) return 0;
    else {
        return arr[n-1] + (sum(arr, n-1));
    }
}
```

2. Rewrite the recursive sum function to only sum odd numbers within the array.

```
int sum(int* arr, int n) {
    if (n == 0) return 0;
    else if (n % 2) {
        return arr[n-1] + (sum(arr, n-1));
    } else {
        return sum(arr, n-1);
    }
}
```

3. Write a recursive function that can find the minimum of a given array.

```
int arr_min(int* arr, int n) {
    if (n == 1) return arr[n-1];
    int func_val = arr_min(arr, n-1);
```

```
    return arr[n-1] < func_val ? arr[n-1] : func_val;
}
```

4. Reverse the elements of an array in place. Matching the following function signature:

```
void reverse_array(int* arr, int n);
```

```
void reverse_array(int* arr, int n)
{
    if (n > 1)
    {
        //swap is not an actual function, have students write it
        swap(arr, arr + n - 1);
        reverse_array(arr + 1, n - 2);
    }
}
```

5. Write a function to print triangles to `std::cout` that takes three positive integers: a, b, c as input. The function should print the `+` character a times, then $a+c$ times, then $a+c+c$ times, and so on. This pattern should repeat until the line is b characters long. At that point, the pattern is repeated backwards. For example calling `draw_triangle(4, 7, 1)` will output: (where the dollar symbol is the bash command prompt)

```
++++
+++++
++++++
+++++++
+++++++
++++++
+++++
++++
```

```
void draw_triangle(unsigned a, unsigned b, unsigned c)
{
    if(a < b)
    {
        for(unsigned i = 0; i < a; i++)
            std::cout << '+';
        std::cout << '\n';
        draw_triangle(a + c, b, c);
        for(unsigned i = 0; i < a; i++)
            std::cout << '+';
        std::cout << '\n';
    }
}
```

6. Recursively multiply two numbers together, *without using the * operator*. Matching the following function signature:

```
int multiply(int a, int b);
```

```
int multiply(int a, int b)
{
    if(b == 0) return 0;
    else if(b < 0) return a + multiply(a, b + 1);
    else return a + multiply(a, b - 1);
}
```

7. Write a function that returns true if a character array is a palindrome and false if it is not. Match the following function signature:

```
bool palindrome(char *a, int length);
```

```
bool palindrome(char *a, int length)
{
    if(length <= 1) return true;
    if(a[0] != a[length - 1]) return false;
    return palindrome(a + 1, length - 2);
}
```

8. Write a recursive function that returns the nth member of the Fibonacci series, with elements 0 and 1 being 1 and 1 (so the series starts 1, 1, 2, 3, 5, 8, 13, ...). Match the following signature:

   `unsigned fibSeries(unsigned n);`

```
unsigned fibSeries(unsigned n)
{
    if(n == 1 || n == 0) return 1;
    return fibSeries(n - 1) + fibSeries(n - 2);
}
```

9. For both insertion and selection sort, describe if the algorithm is stable and if not give an example array that shows the unstable behavior.

   Insertion sort: Stable
   Selection sort: Unstable (unless implemented with extra memory or using non-standard insertion method like swapping more than 2 elements per insertion); [1,3,3,2]