Quality Code, Optimization, and Problem Solving

Code Camp Week 6

Readability

The level to which a person can read and understand code that they did not write

Why is **readable** code important?

- Working on code with a coworker
- Finishing a project with a partner
- Reading documentation about an API
- Understanding standard library functions

All of these situations require readable code…

Hall of Shame vs. Fame

```
31
32  int function1 (string a[], int n) {
33      if(n<0){ return -1; }
34      for(int eger = 0; eger < n; eger++){
35          if(eger == n/2){ break; }
36          string blah = a[eger];
37          a[eger] = a[n-(eger+1)];
38          a[n-(eger+1)] = blah;
39      }
40      return n;
41  }
42
```

```
43  // reverses array of strings, a
44  // returns the length of the array or -1 on error
45  int reverse_array (string a[], int n) {
46      if (n < 0) {
47          return -1;
48      }
49
50      int lower = 0;
51      int higher = n - 1;
52      string temp = "";
53      while (lower < higher) {
54          temp = a[lower];
55          a[lower++] = a[higher];
56          a[higher--] = temp;
57      }
58      return n;
59  }
```

Tips for Improving Readability

- Choose variable and function names that describe them
  - Ex. int counter = 0, reverse_string(), find_maximum(), sort()
- Use indentation and spacing properly
- Utilize comments when needed
- Write a README

Optimization Tips

- Don’t worry in CS 31, but it’s never too early to think about it
- Most optimization happens in your choice of algorithm
  - E.g. In a set of integers find a grouping of 3 integers which maximizes the sum of the 3 integers
- Roughly speaking, the following is true:
  - Using operations (+, -, comparisons) increases your time complexity
  - Using variables increases what is called space complexity

Code Motion

- Moving code around to (hopefully) improve performance

```
for (int i = 0; i < n; ++i) {
    x = y + z;
    a[i] = 6 * i + x * x;
}
```

```
x = y + z;
t1 = x * x;
for (int i = 0; i < n; ++i) {
    a[i] = 6 * i + t1;
}
```

In-Place Manipulation

- Doing things like string reversals, or list rotating can be done in-place and utilize much less memory when doing so

A Note Regarding Optimization

- Don’t worry about in CS 31
- In general, don’t worry about it too much unless it’s a problem
  - E.g. something is taking too long, using too much memory, etc.
- That being said, optimizing as you go is good because you can avoid the problem of going back as long as possible

- DO NOT attempt to optimize without profiling

What is profiling?

- Method of determining which parts of your code are utilizing the most resources
- Can help you identify performance bottlenecks
- This is where you should seek to optimize

Real-World Practice Problem #1

- Write a function which can verify the solution to a Sudoku puzzle

2	9	5	7	4	3	8	6	1
4	3	1	8	6	5	9	2	7
8	7	6	1	9	2	5	4	3
3	8	7	4	5	9	2	1	6
6	1	2	3	8	7	4	9	5
5	4	9	2	1	6	7	3	8
7	6	3	5	3	4	1	8	9
9	2	8	6	7	1	3	5	4
1	5	4	9	3	8	6	7	2

Real-World Practice Solution #1

- Write a function which can verify the solution to a Sudoku puzzle

```
public boolean isValidSudoku(char[][] board) {
    for (int i = 0; i < 9; i++) {
        if (!isValid(board, i, 0, i + 1, 9)) {
            return false;
        }
        if (!isValid(board, 0, i, 9, i + 1)) {
            return false;
        }
    }
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (!isValid(board, i * 3, j * 3, i * 3 + 3, j * 3 + 3)) {
                return false;
            }
        }
    }
    return true;
}
```

```
public boolean isValid(char[][] board, int x, int y, int x0, int y0) {
    int[] set = {0, 0, 0, 0, 0, 0, 0, 0, 0};
    for (int i = x; i < x0; i++) {
        for (int j = y; j < y0; j++) {
            char ch = board[i][j];
            if (ch != '.') {
                if (set[ch - '1']++ > 0) {
                    return false;
                }
            }
        }
    }
    return true;
}
```

Real-World Practice Problem #2

- Suppose we have a high speed online game that people are constantly checking in at. We want to be able to understand the age makeup of these people. Design a function with the following specification:
  - Given: An array of 1000 integers representing the ages of 1000 players
  - Provide: The number of players of each age.