HW1 Solution EE538 - Computing Principles for Electrical Engineers

- Please clone the repository, edit README.md to answer the questions, and fill up functions to finish the homework.
- For non-coding questions, fill out the answers below each question. Please write your answer there.
- For coding questions, please make sure that your code can run `bazel run/test`. In this homework, you will need to fill up cpplib.cc and tests in tests. **Do Not change or modify any given function names and input or output formats in both cpplib.cc and tests in tests. Unexpected changes will result in zero credit.**
- For coding questions, there is a black box test for each question. All points are only based on passing the test cases or not (i.e. we don't grade your work by your source code). So, try to do comprehensive testing before your final submission.
- For submission, please push your answers to Github before the deadline.
- Deadline: **Friday, June 3rd by 12 pm (noon)**
- Total: 130 points. 100 points is considered full credit.

Question 1 (10 Points. Easy)

Create an account on GitHub and Stack Overflow and paste the link to your profile.

GitHub profile link: https://github.com/hongshuochen

Stack Overflow profile link: https://stackoverflow.com/users/14155623/hong-shuo-chen?tab=profile

Question 2 (20 Points. Medium)

Write a function called **FindMedian**, which returns the median of the inputs of a vector of integers. Use the steps the following steps:

1. Clearly specify input and output.
2. Write some example input and outputs. Try to cover corner cases. Feel free to make reasonable assumptions for the corner cases.
3. Implement your algorithm.
4. Write several unit tests that cover all corner cases, and test your algorithm using bazel. You will need to only submit your algorithm implementation and the unit tests.

One possible solution:

```
cpplib.h
float FindMedian(std::vector<int> &input);

cpplib.cc
// It is fine to define return type as double
float CPPLib::FindMedian(std::vector<int> &input){
    size_t vecSize = input.size();
    if(!vecSize) return -1;
```

```
    std::sort(input.begin(), input.end());
    // if input vector is of odd length, median is the middle element
after sorting
    if(vecSize % 2){
        return input[vecSize/2];
    }
    // if input vector is of even length, median is the avg. of the two
elements
    // in the middle after sorting
    else if(!(vecSize % 2)){
        return (input[vecSize/2 - 1] + input[vecSize/2]) / 2.0;
    }
  }
```

Question 3 (20 Points. Medium)

Compute the worst case run-time complexity of the below functions. Please provide both the computation process and final result for full credit.

Rubric: each problem is 5 points(1 for computation process).

```
void Example1(int n) {
   int i = 1;
   while (i < n) {
       i *= 2;
   }
}
```

Answer: O(logn)

```
void Example2(int n) {
   int count = 0;
   for (int i = 1; i <= n; i = i * 2) {
       for (int j = 1; j <= n; j++) {
           for (int k = 1; k <= n; k = k * 3) {
               count++;
           }
       }
   }
}
```

Answer: O(n*log(n)^2)

```
void Example3(int n) {
   int count = 0;
   for (int i = 0; i < n; i++)
       for (int j = i; j < i*i*i; j++)
```

```
            std::cout<<"*";
}
```

Hint: Note the `j < i*i*i` in the inner loop and compute the cube sequence to get the final result.

Answer: O(n^4)

```
int Example4(int n) {
   int count = 0;
   for (int i = 1; i < n; i *= 3) {
      for (int j = n; j > 0; j /= 3) {
          for (int k = 0; k < j; k++) {
              count += 1;
          }
      }
   }
   return count;
}
```

Hint: Note the `i /= 3` in the outer loop and compute the geometric sequence to get the final result.

Answer: O(nlogn)

Question 4 (10 Points. Easy)

Write a function **RandomCase** to manipulate the input string so that the case of each character of the string is selected randomly. Write some tests using GTest for for your function in `tests/q4_student_test.cc`.

Example input & output:

```
Input: "Random"
Possible Outputs: "rAnDom" or "RAndOM" or "rANdOm"
```

One possible solution

```
cpplib.h
std::string RandomCase(std::string &inputs);

cpplib.cc
std::string CPPLib::RandomCase(std::string &inputs) {
    if (!inputs.size()) return "Empty input";
    std::string result = "";
    for(int i = 0; i < inputs.size(); i++){
        int k = rand() % 2;
        if (k == 0){
            char l = inputs[i];
            result = result + std::string(1, std::toupper(l));
```

```
        }else{
            char l = inputs[i];
            result = result + std::string(1, std::tolower(l));
        }
    }
    return result;
}
```

Question 5 (15 Points. Easy)

Write a simple function `std::string CPPLib::PrintIntro()` in cpplib.cc which takes three strings: your first name, last name, and your programming experiences, and returns a string which is: Hi, my name is " {first_name} {last_name}, and my programming experiences are: {experiences}.". Write some tests using GTest for your function in `tests/q5_student_test.cc`. We will run your code and see your printout!

Please create your test cases and run the following command to verify the functionality of your program.

```
bazel test tests:q5_student_test
```

One possible solution:

```
cpplib.h
std::string PrintIntro(std::string &firstName, std::string &lastName,
std::string &programExp);

cpplib.cc
std::string CPPLib::PrintIntro(std::string &firstName, std::string
&lastName, std::string &programExp) {
    std::string result = "Hi, my name is " + \
                         firstName + " " + lastName + \
                         ", and my programming experiences are: " +
programExp;
    return result;
}
```

Question 6 (25 Points. Medium)

Write a function `std::vector<int> CPPLib::Flatten3DVector(const std::vector< std::vector< std::vector<int> > > &input)` in cpplib.cc to flatten a 3D vector into a 1D vector.

Example:
Input: inputs = [[[1, 2], [3, 4]], [[5], [6], []], [[7, 8]]].
Output: result = [1, 2, 3, 4, 5, 6, 7, 8].

Write several tests using GTest for your function in tests/q6_student_test.cc.
(Hint: include cases with empty vectors)

Please create your test cases and run the following command to verify the functionality of your program.

```
bazel test tests:q6_student_test
```

What is the worst case runtime complexity of your implementation?

Solution:

```
// Question 6 Flatten a 3D vector
std::vector<int> CPPLib::Flatten3DVector(const
std::vector<std::vector<std::vector<int> > > &input) {
    std::vector<int> res;
    if (input.empty()){
        return res;
    }
    for(auto vec1: input){
        if(vec1.empty()){
            continue;
        }
        for(auto vec2: vec1){
            if(vec2.empty()){
                continue;
            }
            for(auto i: vec2){
                res.push_back(i);
            }
        }
    }
    return res;
}
```

Question 7 (30 Points. Medium)

Write a function `int CPPLib::climbStairs(int n)` in cpplib.cc using recursion to find how many distinct ways you can climb to the top. Your function should accept positive numbers less than 45 and return the number of ways. Further, write several tests using GTest for your function in tests/q7_student_test.cc and compute the time complexity of your implementation.

Rules of the climb stairs
You are climbing a staircase. Each time you can either climb 1 or 2 steps. It takes n steps to reach the top.

If there are 0 stairs, there is 0 way to the top. For negative input, please return -1.

For example, if the stairs number is 4(n = 4), it should have 5 ways to the top.
1 + 1 + 1 + 1
1 + 1 + 2
1 + 2 + 1

2 + 1 + 1
2 + 2

Please create your test cases and run the following command to verify the functionality of your program.

```
bazel test tests:q7_student_test
```

Hint: Try to write a recursive relationship. What is the complexity of your implementation?

Answer: O(2^n)

```
int CPPLib::climbStairs(int n){
    if(n < 0) return -1;
    if(n == 0) return 0;
    if(n == 1) return 1;
    if(n == 2) return 2;
    return climbStairs(n - 1) + climbStairs(n - 2);
}
```