EE599

HW1

Student Name: Shi-Lin Chen

Student ID: 2991911997

Student Email: shilinch@usc.edu

1. Proof by contradiction

we return result as a maximum value

∴ result is either

① not an element of inputs

② has an element input[i] such that
max < input[i]

→ we see that result is assigned to the elements of input
so (1) is contradicted

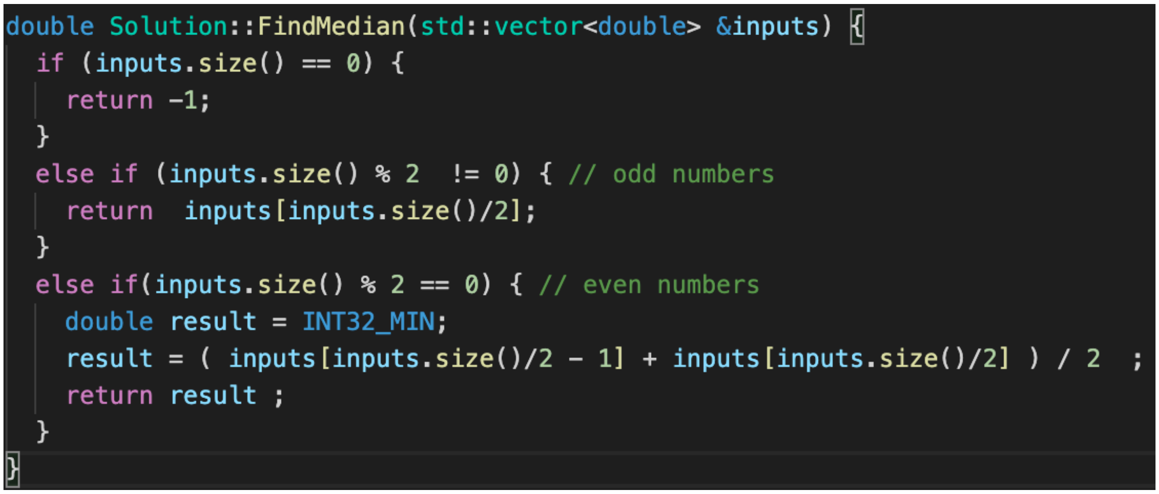
After n-iteration of the for-loop, result ≥ input[n], result only increases which contradicts ②

∴ result is always the maximum value

2. Github: https://github.com/linchen1010/

Stackoverflow: https://stackoverflow.com/users/12746413/linchen1010

3.

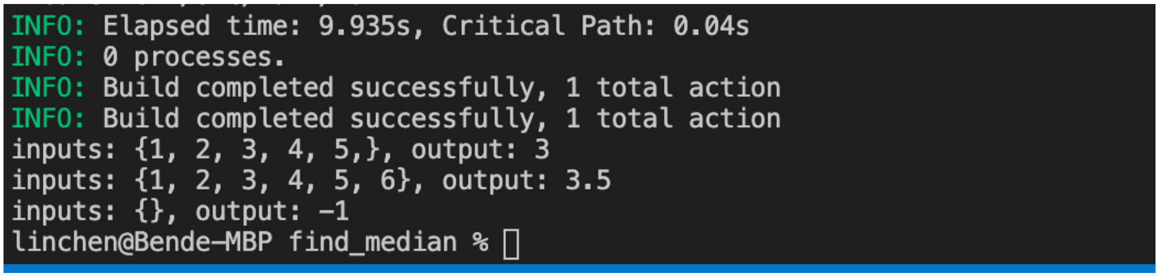


```
double Solution::FindMedian(std::vector<double> &inputs) {
  if (inputs.size() == 0) {
    return -1;
  }
  else if (inputs.size() % 2  != 0) { // odd numbers
    return  inputs[inputs.size()/2];
  }
  else if(inputs.size() % 2 == 0) { // even numbers
    double result = INT32_MIN;
    result = ( inputs[inputs.size()/2 - 1] + inputs[inputs.size()/2] ) / 2  ;
    return result ;
  }
}
```

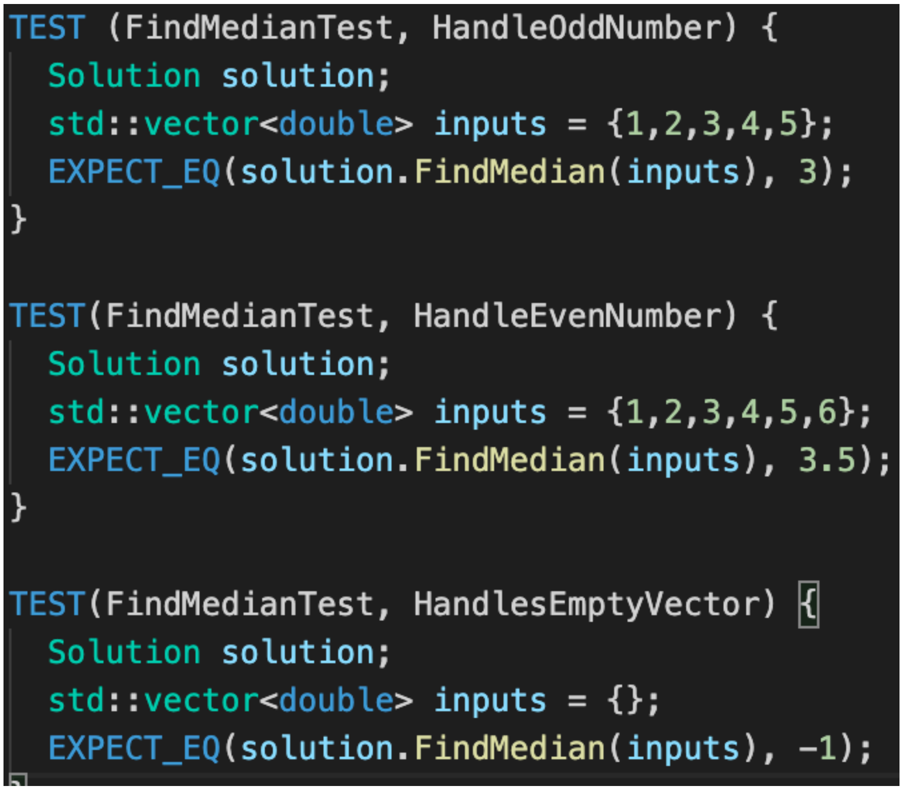
Runtime:

There is no loop in the function, therefore, T(n) = O(1)

Result:



```
INFO: Elapsed time: 9.935s, Critical Path: 0.04s
INFO: 0 processes.
INFO: Build completed successfully, 1 total action
INFO: Build completed successfully, 1 total action
inputs: {1, 2, 3, 4, 5,}, output: 3
inputs: {1, 2, 3, 4, 5, 6}, output: 3.5
inputs: {}, output: -1
linchen@Bende-MBP find_median %
```



```
TEST (FindMedianTest, HandleOddNumber) {
  Solution solution;
  std::vector<double> inputs = {1,2,3,4,5};
  EXPECT_EQ(solution.FindMedian(inputs), 3);
}

TEST(FindMedianTest, HandleEvenNumber) {
  Solution solution;
  std::vector<double> inputs = {1,2,3,4,5,6};
  EXPECT_EQ(solution.FindMedian(inputs), 3.5);
}

TEST(FindMedianTest, HandlesEmptyVector) {
  Solution solution;
  std::vector<double> inputs = {};
  EXPECT_EQ(solution.FindMedian(inputs), -1);
```

Test-case include empty set, odd-number set and even-number set!

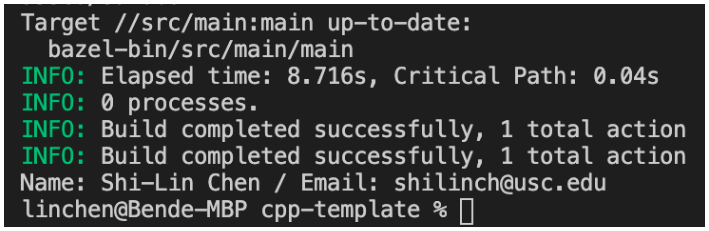
4.

-f >> --force

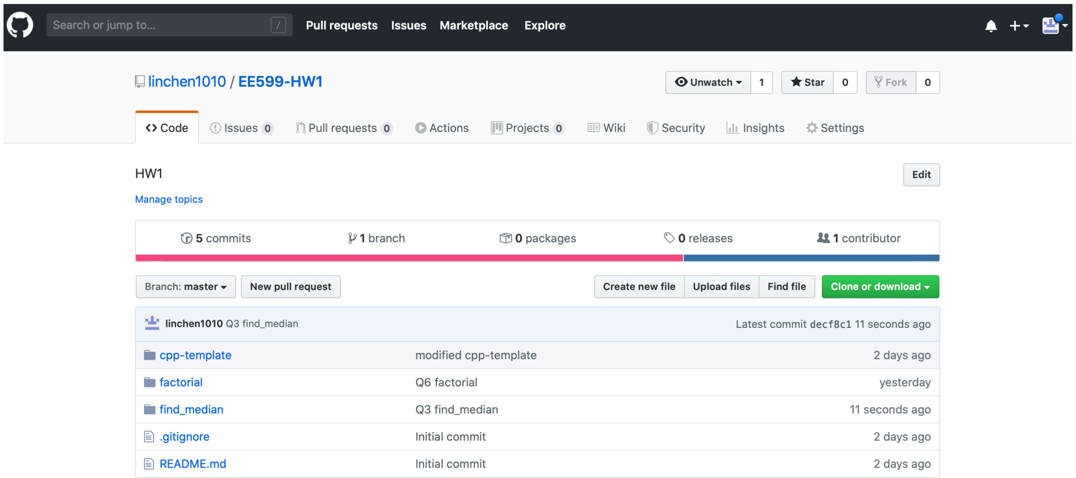
Override the up-to-date check

-r

Allow recursive removal when a leading directory name is given



```
Target //src/main:main up-to-date:
  bazel-bin/src/main/main
INFO: Elapsed time: 8.716s, Critical Path: 0.04s
INFO: 0 processes.
INFO: Build completed successfully, 1 total action
INFO: Build completed successfully, 1 total action
Name: Shi-Lin Chen / Email: shilinch@usc.edu
linchen@Bende-MBP cpp-template %
```



linchen1010 / EE599-HW1

HW1

5 commits | 1 branch | 0 packages | 0 releases | 1 contributor

linchen1010 Q3 find_median — Latest commit decf8c1 11 seconds ago

cpp-template	modified cpp-template	2 days ago
factorial	Q6 factorial	yesterday
find_median	Q3 find_median	11 seconds ago
.gitignore	Initial commit	2 days ago
README.md	Initial commit	2 days ago

5. Time complexity

1.

The outter for-loop is $O(\log n)$

The inner for-loop is controled by $\log n$

$\therefore$ The time complexity should be $O(\log 2n)$

2. $\log n$

since the while-loop is controled by n

1st iteration $\Rightarrow$ $x = n/2$

2nd iteration $\Rightarrow$ $x = n/4$

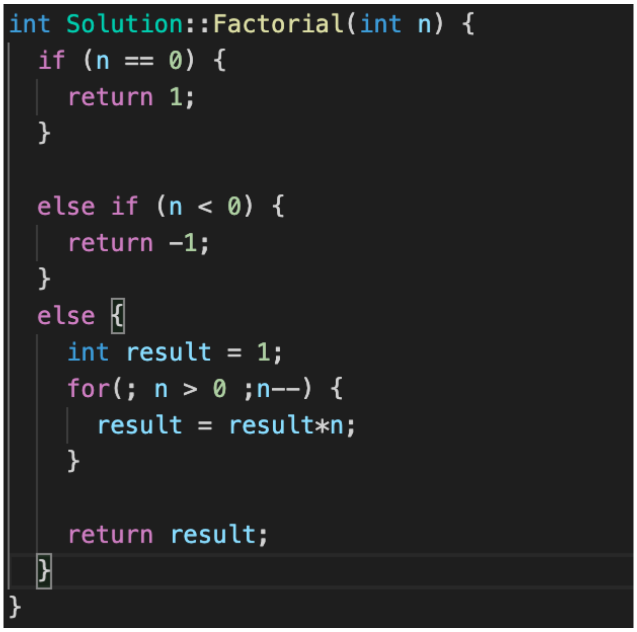
$\vdots$

K 〃 $\Rightarrow$ $x = n/2^K = 1$

$\therefore K = \log_2 n$ $\therefore O(\log n)$

6.

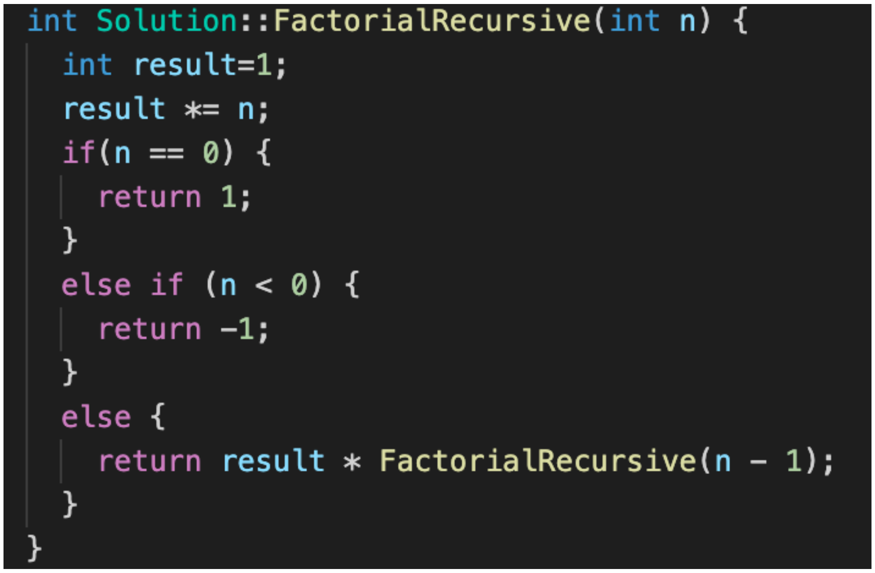
Code is included in the folder



```
int Solution::Factorial(int n) {
  if (n == 0) {
    return 1;
  }

  else if (n < 0) {
    return -1;
  }
  else {
    int result = 1;
    for(; n > 0 ;n--) {
      result = result*n;
    }

    return result;
  }
}
```



```
int Solution::FactorialRecursive(int n) {
  int result=1;
  result *= n;
  if(n == 0) {
    return 1;
  }
  else if (n < 0) {
    return -1;
  }
  else {
    return result * FactorialRecursive(n - 1);
  }
}
```

6.

Non-recursive

since there is one for-loop controled by n

∴ the time complexity should be O(n)

Recursive:

it only does constant time and then the function return itself

∴ the time complexity is O(1)

6.5

we can use proof by introduction.

1. Assume n = k is true, and k > 0

   By the function code I wrote.

   Factorial(k) = k * (k-1) * (k-2) …. 2 * 1 = k! = Factorial(n)

2. Induction part, we want to show n = k+1 is also true

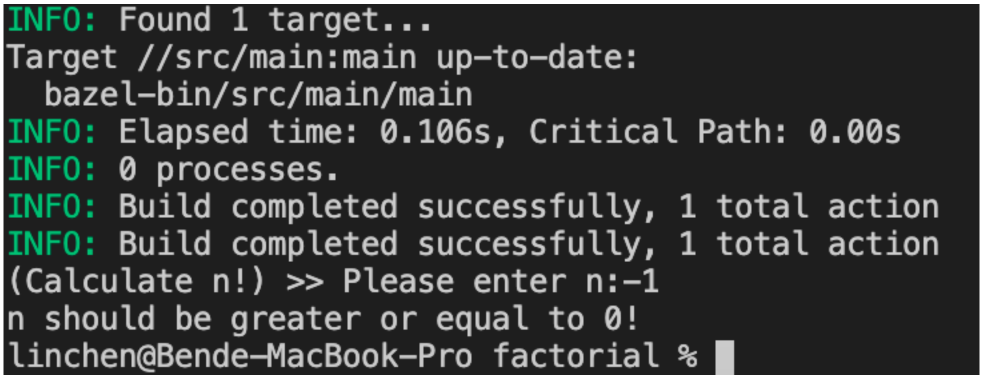
   If n = k + 1

   Factorial(k+1) = (k+1) * k * (k-1) * (k-2) * …. * 2 * 1 = (k+1)! = Factorial(n)

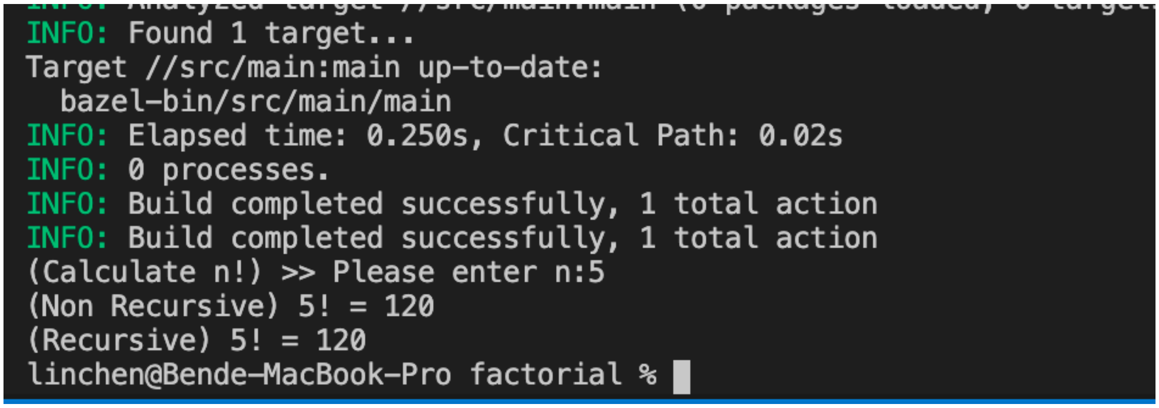
Therefore,

Factorial(n) = n!   QED

Result:



```
INFO: Found 1 target...
Target //src/main:main up-to-date:
  bazel-bin/src/main/main
INFO: Elapsed time: 0.106s, Critical Path: 0.00s
INFO: 0 processes.
INFO: Build completed successfully, 1 total action
INFO: Build completed successfully, 1 total action
(Calculate n!) >> Please enter n:-1
n should be greater or equal to 0!
linchen@Bende-MacBook-Pro factorial %
```



```
INFO: Analyzed target //src/main:main (0 packages loaded, 0 target
INFO: Found 1 target...
Target //src/main:main up-to-date:
  bazel-bin/src/main/main
INFO: Elapsed time: 0.250s, Critical Path: 0.02s
INFO: 0 processes.
INFO: Build completed successfully, 1 total action
INFO: Build completed successfully, 1 total action
(Calculate n!) >> Please enter n:5
(Non Recursive) 5! = 120
(Recursive) 5! = 120
linchen@Bende-MacBook-Pro factorial %
```

Test cases include n < 0, n=0, n > 0