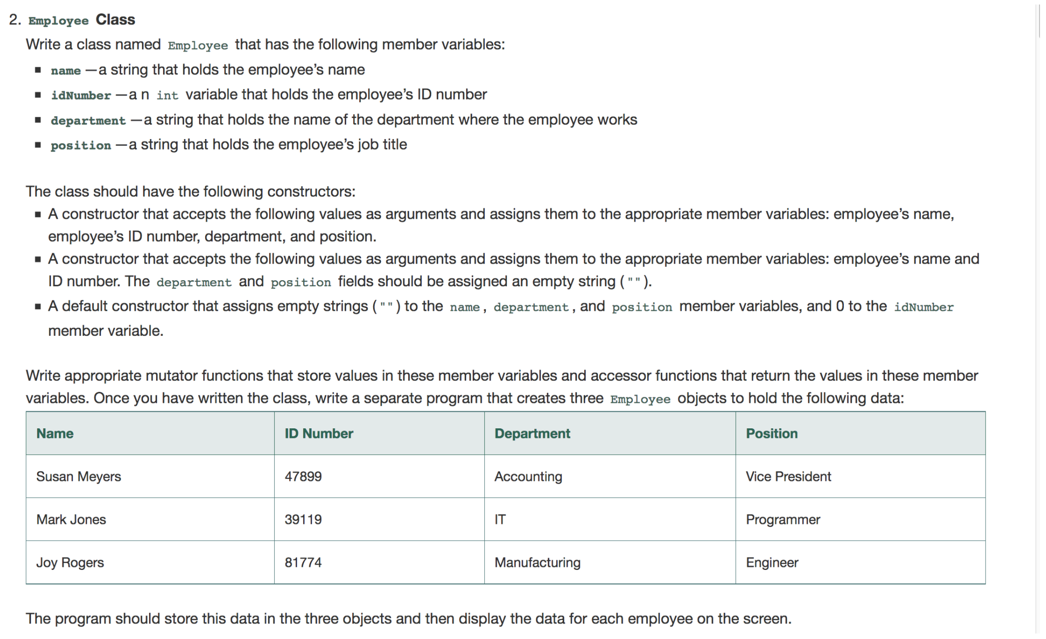
COMSC-200
Lab 1

Ryan Jacoby

6 September 2020

1 Employee



2. `Employee` **Class**

Write a class named `Employee` that has the following member variables:

- `name` —a string that holds the employee's name
- `idNumber` —a n `int` variable that holds the employee's ID number
- `department` —a string that holds the name of the department where the employee works
- `position` —a string that holds the employee's job title

The class should have the following constructors:

- A constructor that accepts the following values as arguments and assigns them to the appropriate member variables: employee's name, employee's ID number, department, and position.
- A constructor that accepts the following values as arguments and assigns them to the appropriate member variables: employee's name and ID number. The `department` and `position` fields should be assigned an empty string (`""`).
- A default constructor that assigns empty strings (`""`) to the `name`, `department`, and `position` member variables, and 0 to the `idNumber` member variable.

Write appropriate mutator functions that store values in these member variables and accessor functions that return the values in these member variables. Once you have written the class, write a separate program that creates three `Employee` objects to hold the following data:

Name	ID Number	Department	Position
Susan Meyers	47899	Accounting	Vice President
Mark Jones	39119	IT	Programmer
Joy Rogers	81774	Manufacturing	Engineer

The program should store this data in the three objects and then display the data for each employee on the screen.

```cpp
// Ryan Jacoby

#include<iostream>

#include"Employee.h"

using namespace std;

int main() {
    Employee employees[] = {Employee("Susan Meyers", 47899, "Accounting", "Vice
President"),
                            Employee("Mark Jones", 39119, "IT\t", "Programmer"),
                            Employee("Joy Rodgers", 81774, "Manufacturing", "
Engineer")};


    cout << "Name\t\tID Number\tDepartment\tPosition\n";
    for(int i = 0; i < 3; i++)
```

```
        cout << employees[i].getName() << '\t' << employees[i].getIdNumber() << "\
    t\t" << employees[i].getDepartment() << '\t' << employees[i].getPosition() << '
    \n';

    return 0;
}
```

Listing 1: main.cpp

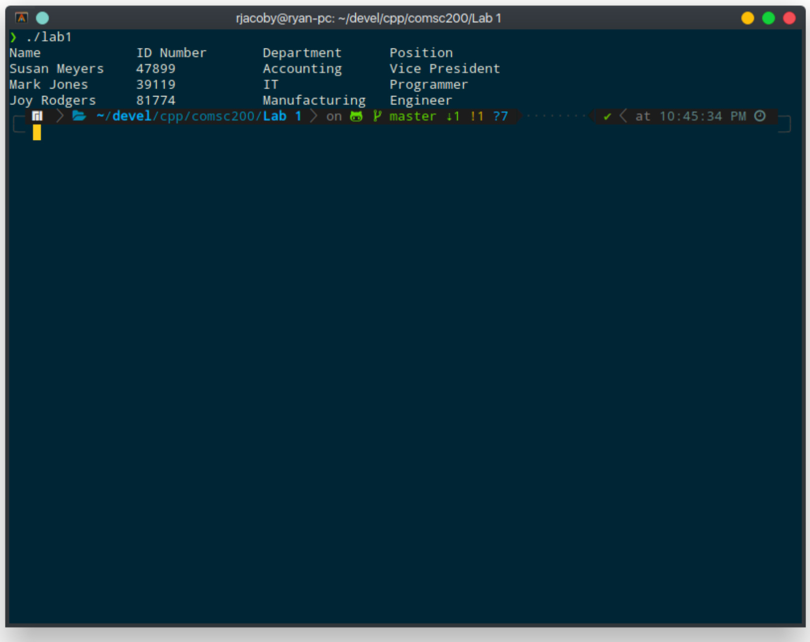
```
// Ryan Jacoby

#ifndef Employee_h
#define Employee_h

using namespace std;

class Employee {
private:
    string name;
    int idNumber;
    string department;
    string position;
public:
    Employee();
    Employee(string name, int idNumber);
    Employee(string name, int idNumber, string department, string position);

    string getName() { return this->name; }
    int getIdNumber() { return this->idNumber; }
    string getDepartment() { return this->department; }
    string getPosition() { return this->position; }

    void setName(string name) { this->name = name; }
    void setIdNumber(int idNumber) { this-> idNumber = idNumber; }
    void setDepartment(string department) { this->department = department; }
    void setPosition( string position ) { this->position = position; }
};

#endif
```

Listing 2: Employee.h

```
// Ryan Jacoby

#include<string>

#include"Employee.h"

using namespace std;

Employee::Employee() {
    this->name = "";
    this->idNumber = 0;
    this->department = "";
    this->position = "";
}

Employee::Employee(string name, int idNumber) {
    this->name = name;
```

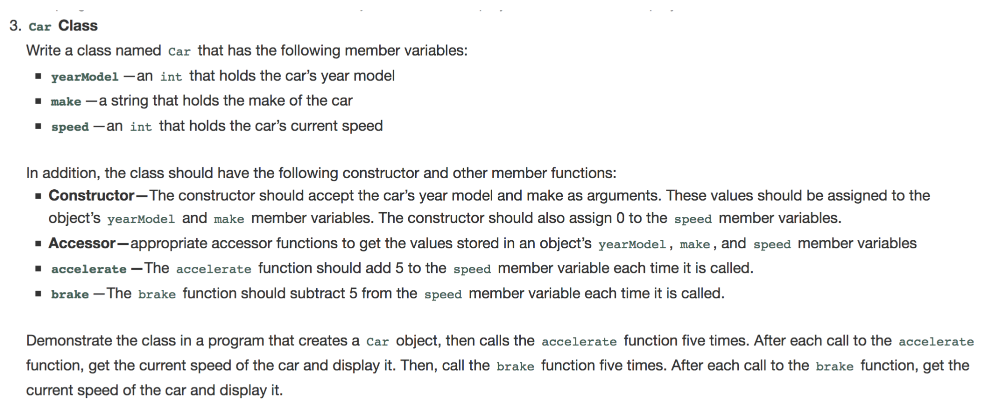
```cpp
    this->idNumber = idNumber;
    this->department = "";
    this->position = "";
}

Employee::Employee(string name, int idNumber, string department, string position)
    {
    this->name = name;
    this->idNumber = idNumber;
    this->department = department;
    this->position = position;
}
```

Listing 3: EmployeeImp.cpp



```
rjacoby@ryan-pc: ~/devel/cpp/comsc200/Lab 1
❯ ./lab1
Name            ID Number       Department      Position
Susan Meyers    47899           Accounting      Vice President
Mark Jones      39119           IT              Programmer
Joy Rodgers     81774           Manufacturing   Engineer
 ~/devel/cpp/comsc200/Lab 1 on  master ⇣1 !1 ?7 ········ ✔ at 10:45:34 PM
```

2 Car



3. `Car` **Class**

Write a class named `Car` that has the following member variables:

- **`yearModel`** —an `int` that holds the car's year model
- **`make`** —a string that holds the make of the car
- **`speed`** —an `int` that holds the car's current speed

In addition, the class should have the following constructor and other member functions:

- **Constructor—**The constructor should accept the car's year model and make as arguments. These values should be assigned to the object's `yearModel` and `make` member variables. The constructor should also assign 0 to the `speed` member variables.
- **Accessor—**appropriate accessor functions to get the values stored in an object's `yearModel`, `make`, and `speed` member variables
- **`accelerate`** —The `accelerate` function should add 5 to the `speed` member variable each time it is called.
- **`brake`** —The `brake` function should subtract 5 from the `speed` member variable each time it is called.

Demonstrate the class in a program that creates a `Car` object, then calls the `accelerate` function five times. After each call to the `accelerate` function, get the current speed of the car and display it. Then, call the `brake` function five times. After each call to the `brake` function, get the current speed of the car and display it.

```cpp
// Ryan Jacoby
```

```cpp

#include<iostream>

#include"Car.h"

using namespace std;

int main() {
    Car c = Car(2020, "Virtual Car");

    cout << "New car created\n Year: " << c.getYearModel() << "\nModel: " << c.
   getMake() << '\n';

    for(int i = 0; i < 5; i++) {
        cout << "Car current speed: " << c.getSpeed() << '\n';
        c.accelerate();
    }

    cout << "Car current speed: " << c.getSpeed() << '\n';

    for(int i = 0; i < 5; i++) {
        cout << "Car current speed: " << c.getSpeed() << '\n';
        c.brake();
    }

    return 0;
}
```

Listing 4: main.cpp

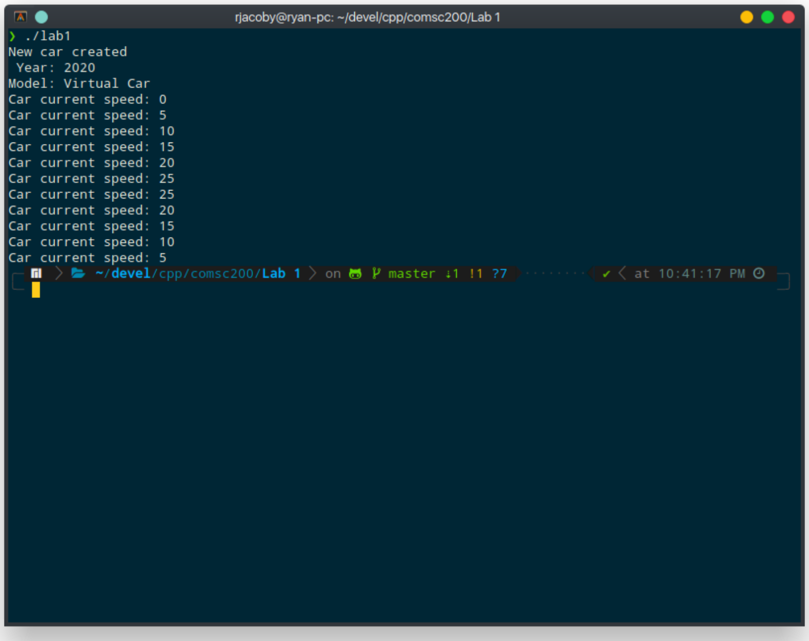
```cpp
// Ryan Jacoby

#ifndef Car_h
#define Car_h

using namespace std;

class Car {
private:
    int yearModel;
    string make;
    int speed;
public:
    Car(int yearModel, string make);

    int getYearModel() { return this->yearModel; }
    string getMake() { return this->make; }
    int getSpeed() { return this->speed; }

    void accelerate();
    void brake();
};

#endif
```

Listing 5: Car.h

```cpp
// Ryan Jacoby

```

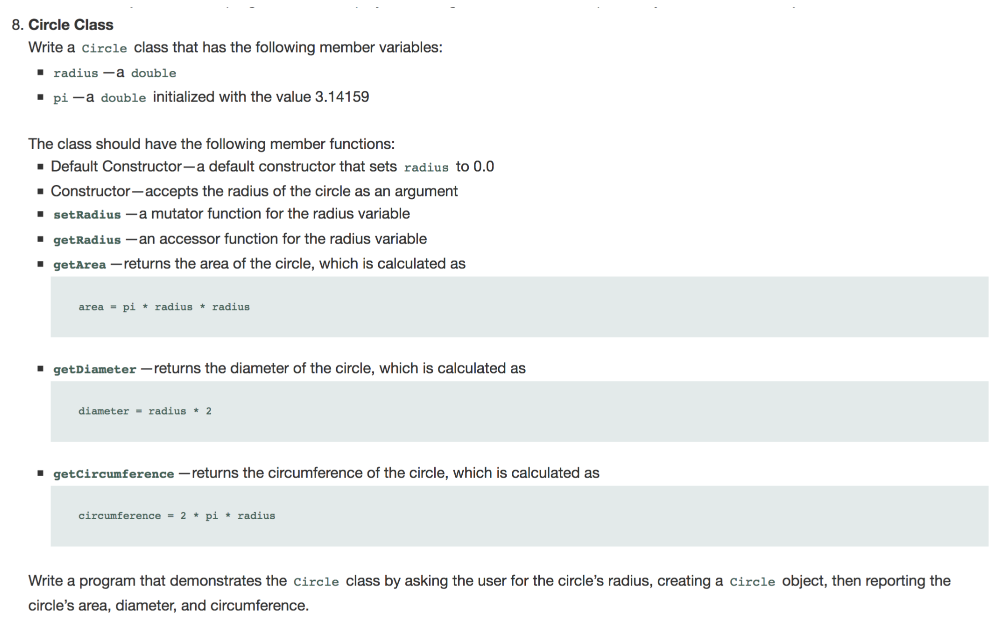
```cpp
#include<string>

#include"Car.h"

using namespace std;

Car::Car(int yearModel, string make) {
    this->yearModel = yearModel;
    this->make = make;
    this->speed = 0;
}

void Car::accelerate() {
    speed += 5;
}

void Car::brake() {
    speed -= 5;
}
```

Listing 6: CarImp.cpp



```
rjacoby@ryan-pc: ~/devel/cpp/comsc200/Lab 1
❯ ./lab1
New car created
 Year: 2020
Model: Virtual Car
Car current speed: 0
Car current speed: 5
Car current speed: 10
Car current speed: 15
Car current speed: 20
Car current speed: 25
Car current speed: 25
Car current speed: 20
Car current speed: 15
Car current speed: 10
Car current speed: 5
  > ~/devel/cpp/comsc200/Lab 1 > on  master ↓1 !1 ?7 ········ ✓ < at 10:41:17 PM
```

3 Circle



8. **Circle Class**

Write a `Circle` class that has the following member variables:

- `radius` —a `double`
- `pi` —a `double` initialized with the value 3.14159

The class should have the following member functions:

- Default Constructor—a default constructor that sets `radius` to 0.0
- Constructor—accepts the radius of the circle as an argument
- `setRadius` —a mutator function for the radius variable
- `getRadius` —an accessor function for the radius variable
- `getArea` —returns the area of the circle, which is calculated as

```
area = pi * radius * radius
```

- `getDiameter` —returns the diameter of the circle, which is calculated as

```
diameter = radius * 2
```

- `getCircumference` —returns the circumference of the circle, which is calculated as

```
circumference = 2 * pi * radius
```

Write a program that demonstrates the `Circle` class by asking the user for the circle's radius, creating a `Circle` object, then reporting the circle's area, diameter, and circumference.

```cpp
// Ryan Jacoby

#include<iostream>

#include"Circle.h"

using namespace std;

int main() {
    double rad;
    Circle c = Circle();

    cout << "What size circle do you want: ";
    cin >> rad;

    c.setRadius(rad);

    cout << "That circle has an area of " << c.getArea() << ", a diameter of " << c.getDiameter() << ", and a circumference of " << c.getCircumference() << ".\n";

    return 0;
}
```

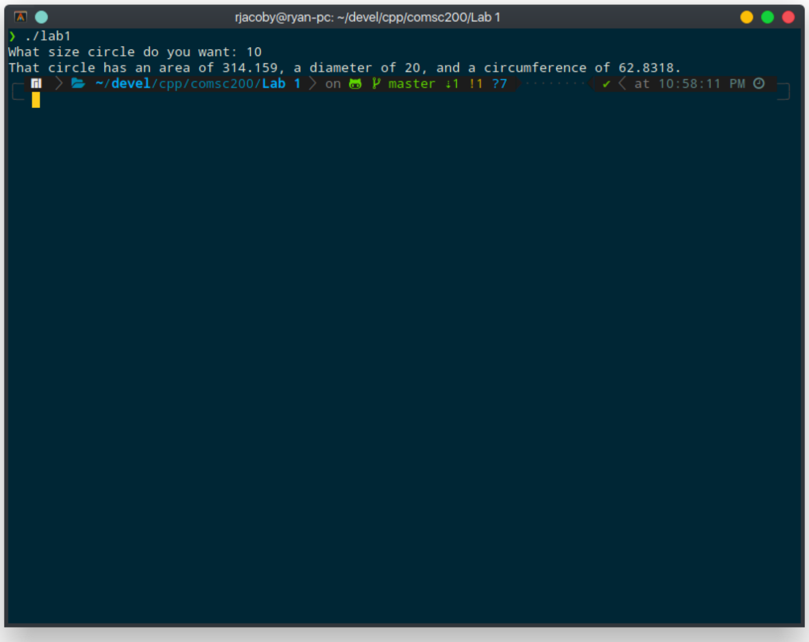
Listing 7: main.cpp

```cpp
// Ryan Jacoby

#ifndef Circle_h
#define Circle_h

class Circle {
private:
    double radius;
    double pi;
public:
    Circle();
    Circle(double radius);

    void setRadius(double radius) { this->radius = radius; }
    double getRadius() { return this->radius; }

    double getArea();
    double getDiameter();
    double getCircumference();
};

#endif
```

Listing 8: Circle.h

```cpp
// Ryan Jacoby

#include"Circle.h"

Circle::Circle() {
    this->radius = 0;
    this->pi = 3.14159;
}

Circle::Circle(double radius) {
    this->radius = radius;
    this->pi = 3.14159;
}

double Circle::getArea() {
    return this->pi * this->radius * this->radius;
}

double Circle::getDiameter() {
    return this->radius * 2;
}

double Circle::getCircumference() {
    return this->pi * this->radius * 2;
}
```

Listing 9: CircleImp.cpp



```
rjacoby@ryan-pc: ~/devel/cpp/comsc200/Lab 1
❯ ./lab1
What size circle do you want: 10
That circle has an area of 314.159, a diameter of 20, and a circumference of 62.8318.
  > ~/devel/cpp/comsc200/Lab 1 > on  master ⇣1 !1 ?7 ········ ✓ < at 10:58:11 PM
```

4 Number Array

10. **Number Array Class**

Design a class that has an array of floating-point numbers. The constructor should accept an integer argument and dynamically allocate the array to hold that many numbers. The destructor should free the memory held by the array. In addition, there should be member functions to perform the following operations:

- Store a number in any element of the array
- Retrieve a number from any element of the array
- Return the highest value stored in the array
- Return the lowest value stored in the array
- Return the average of all the numbers stored in the array

Demonstrate the class in a program.

```cpp
// Ryan Jacoby

#include<iostream>

#include"NumArray.h"

using namespace std;

int main() {
    NumArray n = NumArray(5);

    n.storeNumber(1, 1.5);
    n.storeNumber(0, 2);
    n.storeNumber(2, 2.5);
    n.storeNumber(4, 3);
    n.storeNumber(3, 3.5);

    cout << "1st number: " << n.getNumber(0) << '\n';
    cout << "Highest number: " << n.highestNum() << '\n';
    cout << "Lowest number: " << n.lowestNum() << '\n';
    cout << "Average of numbers: " << n.average() << '\n';

    return 0;
}
```

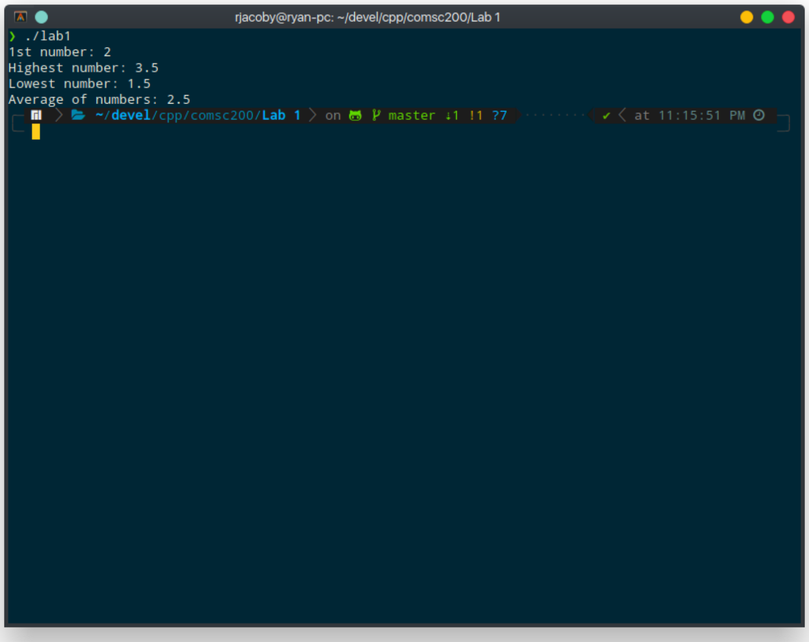
Listing 10: main.cpp

```cpp
// Ryan Jacoby

#ifndef NumArray_h
#define NumArray_h

class NumArray {
private:
    double * nums;
    int elements;
public:
    NumArray(int elements);

    ~NumArray();

    void storeNumber(int pos, double num);
    double getNumber(int pos);
    double highestNum();
    double lowestNum();
    double average();
};

#endif
```

Listing 11: NumArray.h

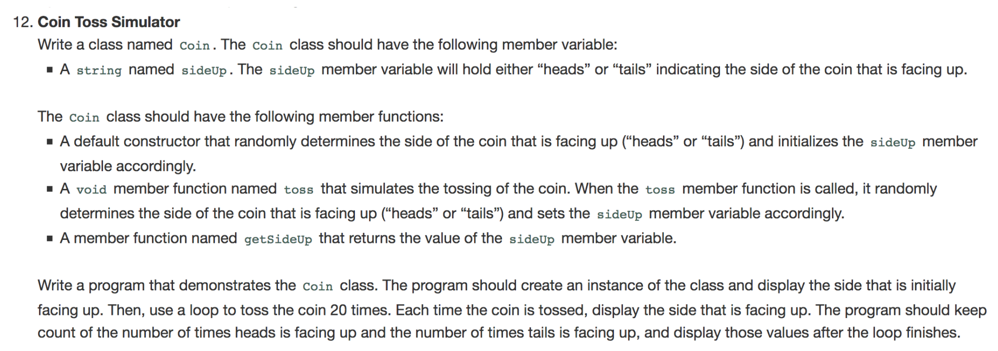
```cpp
// Ryan Jacoby

#include"NumArray.h"

NumArray::NumArray(int elements) {
    this->elements = elements;
    this->nums = new double[elements];
}

NumArray::~NumArray() {
    delete[] this->nums;
}

void NumArray::storeNumber(int pos, double num){
    this->nums[pos] = num;
}

double NumArray::getNumber(int pos) {
    return this->nums[pos];
}

double NumArray::highestNum() {
    double highest = this->nums[0];

    for(int i = 0; i < elements; i++) {
        if(this->nums[i] > highest) highest = this->nums[i];
    }

    return highest;
}

double NumArray::lowestNum() {
    double lowest = this->nums[0];

    for(int i = 0; i < elements; i++) {
        if(this->nums[i] < lowest) lowest = this->nums[i];
    }

    return lowest;
}

double NumArray::average() {
    double sum = 0;

    for(int i = 0; i < elements; i++)
        sum += this->nums[i];

    return sum / elements;
}
```

Listing 12: NumArrayImp.cpp



```
rjacoby@ryan-pc: ~/devel/cpp/comsc200/Lab 1
❯ ./lab1
1st number: 2
Highest number: 3.5
Lowest number: 1.5
Average of numbers: 2.5
  > ~/devel/cpp/comsc200/Lab 1 > on  master ⇣1 !1 ?7 ········· ✔ < at 11:15:51 PM
```

5 Coin



12. **Coin Toss Simulator**

Write a class named `Coin`. The `Coin` class should have the following member variable:

- A `string` named `sideUp`. The `sideUp` member variable will hold either “heads” or “tails” indicating the side of the coin that is facing up.

The `Coin` class should have the following member functions:

- A default constructor that randomly determines the side of the coin that is facing up (“heads” or “tails”) and initializes the `sideUp` member variable accordingly.
- A `void` member function named `toss` that simulates the tossing of the coin. When the `toss` member function is called, it randomly determines the side of the coin that is facing up (“heads” or “tails”) and sets the `sideUp` member variable accordingly.
- A member function named `getSideUp` that returns the value of the `sideUp` member variable.

Write a program that demonstrates the `Coin` class. The program should create an instance of the class and display the side that is initially facing up. Then, use a loop to toss the coin 20 times. Each time the coin is tossed, display the side that is facing up. The program should keep count of the number of times heads is facing up and the number of times tails is facing up, and display those values after the loop finishes.

```cpp
// Ryan Jacoby

#include<iostream>

#include"Coin.h"

using namespace std;

int main() {
    Coin c = Coin();

    for(int i = 0; i < 20; i++) {
        cout << "Coin was " << c.getSideUp() << '\n';
        c.toss();
    }

    return 0;
}
```

Listing 13: main.cpp

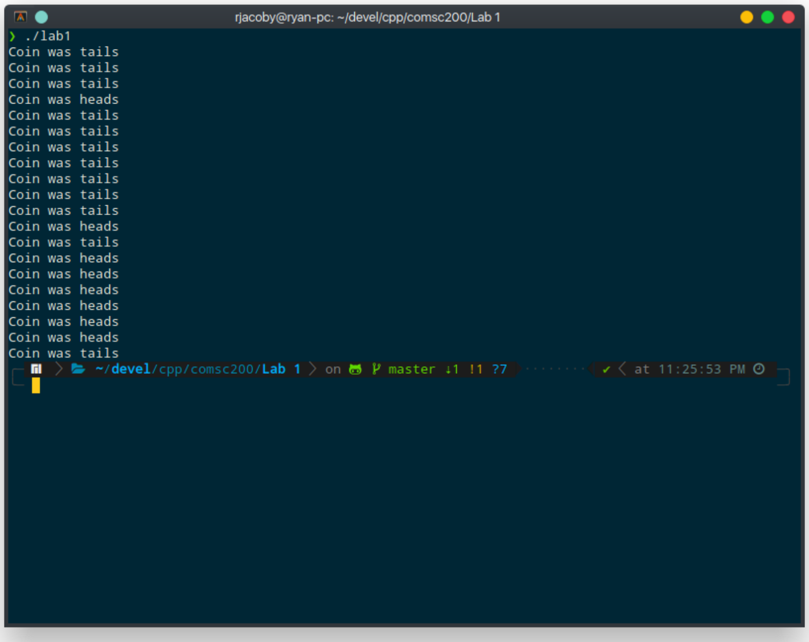
```cpp
// Ryan Jacoby

#ifndef Coin_h
#define Coin_h

using namespace std;

class Coin {
private:
    string sideUp;
public:
    Coin();

    void toss();
    string getSideUp();
};

#endif
```

Listing 14: Coin.h

```cpp
// Ryan Jacoby

#include<string>
```

```cpp

#include<stdlib.h>
#include<time.h>

#include"Coin.h"

using namespace std;

Coin::Coin() {
    srand(time(NULL));

    if(rand() % 2 == 0) this->sideUp = "heads";
    else this->sideUp = "tails";
}

void Coin::toss() {
    if(rand() % 2 == 0) this->sideUp = "heads";
    else this->sideUp = "tails";
}

string Coin::getSideUp() {
    return this->sideUp;
}
```

Listing 15: CoinImp.cpp



```
rjacoby@ryan-pc: ~/devel/cpp/comsc200/Lab 1
❯ ./lab1
Coin was tails
Coin was tails
Coin was tails
Coin was heads
Coin was tails
Coin was tails
Coin was tails
Coin was tails
Coin was tails
Coin was tails
Coin was tails
Coin was heads
Coin was tails
Coin was heads
Coin was heads
Coin was heads
Coin was heads
Coin was heads
Coin was heads
Coin was tails
 > ~/devel/cpp/comsc200/Lab 1 > on  master ⇣1 !1 ?7 ········ ✔ < at 11:25:53 PM
```