Stony Brook University
College of Engineering and Applied Science

ESE 224.L02

Lab 1

Ryan Lin

Professor: Xin Wang

Report Date: 16 September 2023

Problem 1:

Main.cpp

```
int main()
{
    cout << "Creating a point with the default constructor: " << endl;
    Point p1;
    cout << "The point is (" << p1.getX() << ", " << p1.getY() << ")" << endl;

    cout << "Creating a point with the parameterized constructor: " << endl;
    cout << "Enter two coordinates (x, y): ";
    double x, y;
    cin >> x >> y;
    Point p2(x, y);
    cout << "The point is (" << p2.getX() << ", " << p2.getY() << ")" << endl;

    cout << "Change the x coordinate of p2, enter the new coordinate: ";
    cin >> x;
    p2.setX(x);
    cout << "The point is (" << p2.getX() << ", " << p2.getY() << ")" << endl;

    cout << "Change the y coordinate of p2, enter the new coordinate: ";
    cin >> y;
    p2.setY(y);
    cout << "The point is (" << p2.getX() << ", " << p2.getY() << ")" << endl;

    cout << "Compute the point's euclidean distance to origin (0, 0): " << endl;
    double dist = p2.dist2origin();
    cout << "Distance to origin is: " << dist << endl;
    system("pause");
    return 0;
}
```

Point.cpp

```cpp
Point::Point()
{
    xCoord = 0;
    yCoord = 0;
}

Point::Point(double x, double y)
{
    xCoord = x;
    yCoord = y;
}

double Point::getX()
{
    return xCoord;
}

double Point::getY()
{
    return yCoord;
}

void Point::setX(double x)
{
    xCoord = x;
}

void Point::setY(double y)
{
    yCoord = y;
}

double Point::dist2origin()
{
    return sqrt(xCoord * xCoord + yCoord * yCoord);
}
```

Point.h

```
class Point
{
private:
    double xCoord, yCoord;

public:
    Point();
    Point(double x, double y);
    double getX();
    double getY();
    void setX(double x);
    void setY(double y);
    double dist2origin();
};
```

Output:

```
Creating a point with the default constructor:
The point is (0, 0)
Creating a point with the parameterized constructor:
Enter two coordinates (x, y): 1 3
The point is (1, 3)
Change the x coordinate of p2, enter the new coordinate: 4
The point is (4, 3)
Change the y coordinate of p2, enter the new coordinate: 5
The point is (4, 5)
Compute the point's euclidean distance to origin (0, 0):
Distance to origin is: 6.40312
sh: pause: command not found
```

Problem 2:

Main.cpp

```cpp
int main()
{
    Player player1;
    player1.setName("Christian Ronaldo");
    player1.setAge(38);
    player1.setHeight(6.2);
    player1.setNationality("Portugal");

    Player player2;
    player2.setName("Leonel Messi");
    player2.setAge(36);
    player2.setHeight(5.7);
    player2.setNationality("Argentina");

    cout << "Player Information:\n";
    cout << "Name: " << player1.getName() << endl;
    cout << "age: " << player1.getAge() << endl;
    cout << "Height: " << player1.getHeight() << endl;
    cout << "Nationality: " << player1.getNationality() << endl << endl;

    cout << "Player2 Information:\n";
    cout << "Name: " << player2.getName() << endl;
    cout << "age: " << player2.getAge() << endl;
    cout << "Height: " << player2.getHeight() << endl;
    cout << "Nationality: " << player2.getNationality() << endl;

    player2.prediction(player1);

    return 0;
}
```

Player.cpp

```cpp
Player::Player(){
    name = "";
    age = 0;
    height = 0.0;
    nationality = "";
}
string Player::getName() const{
    return name;
}
int Player::getAge() const{
    return age;
}
double Player::getHeight() const{
    return height;}
string Player::getNationality() const{
    return nationality;
}
void Player::setName(const string& newName){
    name = newName;
}
void Player::setAge(int newAge){
    age = newAge;
}
void Player::setHeight(double newHeight){
    height = newHeight;
}
void Player::setNationality(const string& newNationality){
    nationality = newNationality;
}
void Player::prediction(Player p2){
    if (age < p2.age && height > p2.height){
        cout << "The winner is: " << name << endl;
    }
    else if (age > p2.age && height < p2.height){
        cout << "The winner is: " << p2.name << endl;
    }
    else{
        random_device rd;
        default_random_engine generator(rd());
        uniform_real_distribution<double> distribution(0.0, 1.0);
        double random_number = distribution(generator);

        //code to choose random player that wins
        if (random_number < 0.5)
        {
        cout << "The winner is: " << name << endl;
        // cout << random_number;
        }
        else
        {
        cout << "The winner is: " << p2.name << endl;
        // cout << random_number;
        }
    }
}
```

Player.h

```
class Player
{
private:
    string name;
    int age;
    double height;
    string nationality;

public:
    //constructor
    Player();

    //Getter Method
    string getName() const;
    int getAge() const;
    double getHeight() const;
    string getNationality() const;

    //setter Method
    void setName(const string& newName);
    void setAge(int newAge);
    void setHeight(double newHeight);
    void setNationality(const string& newNationality);
    void prediction(Player p2);

};
```

Output:

```
Player Information:
Name: Christian Ronaldo
age: 38
Height: 6.2
Nationality: Portugal

Player2 Information:
Name: Leonel Messi
age: 36
Height: 5.7
Nationality: Argentina
The winner is: Leonel Messi
```

Problem 3:

Main.cpp

```
1   int main(){
2       Student std;
3       std.getDetails();
4       cout << endl;
5       std.displayDetails();
6       cout << endl;
7       Student std2;
8       std2.getDetails();
9       cout << endl;
10      std2.displayDetails();
11      cout << endl;
12      std2.comparison(std);
13
14      return 0;
15  }
```

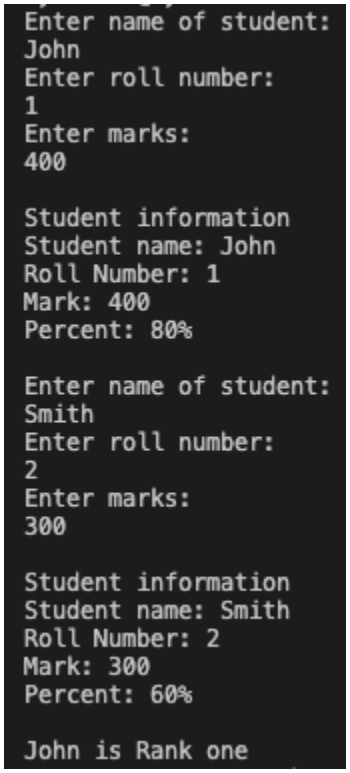
Student.cpp

```cpp
Student::Student(){
    name = "";
    roll_n = 0;
    mark = 0;
    percent = 0.0;
}

void Student::getPercent(){
    int max_mark = 500;
    if (mark == 0)
    {
    cout << "Warning: Mark is 0. Make sure to set a valid mark before calling getPercent." << endl;
    }
    percent = static_cast<double>(mark) / max_mark * 100.00;
}

void Student::comparison(Student s2){
    if(mark > s2.mark){
        cout << name << " is Rank one\n";
    }
    else{
        cout << s2.name << " is Rank one\n";
    }
}

void Student::getDetails(){
    cout << "Enter name of student: \n";
    cin >> name;
    cout << "Enter roll number: \n";
    cin >> roll_n;
    cout << "Enter marks: \n";
    cin >> mark;
    getPercent();
}

void Student::displayDetails(){
    cout << "Student information";
    cout << "\nStudent name: " << name;
    cout << "\nRoll Number: " << roll_n;
    cout << "\nMark: " << mark;
    cout << "\nPercent: " << percent << "%" << endl;
}
```

student.h

```
1   class Student
2   {
3   private:
4       string name;
5       int roll_n;
6       int mark;
7       double percent;
8   public:
9       //constructor
10      Student();
11
12      void getPercent();
13      void comparison(Student s2);
14
15      void getDetails();
16      void displayDetails();
17  };
```

Output:


Enter name of student:
John
Enter roll number:
1
Enter marks:
400
Student information
Student name: John
Roll Number: 1
Mark: 400
Percent: 80%
Enter name of student:
Smith
Enter roll number:
2
Enter marks:
300
Student information
Student name: Smith
Roll Number: 2
Mark: 300
Percent: 60%
John is Rank one

Problem 4:

main.cpp

```cpp
int main(int argc, const char * argv[]){
    cout << "constructing 5 book objects" << endl;
    Book b1(1,"b1", "b1a", "b1g", 1);
    Book b2(2,"b2", "b2a", "b2g", 10);
    Book b3(3,"b3", "b3a", "b3g", 11);
    Book b4(4,"b4", "b4a", "b4g", 100);
    Book b5(5,"b5", "b5a", "b5g", 101);

    b1.display();
    b2.display();
    b3.display();
    b4.display();
    b5.display();
    b5 = b1;
    b5.display();

    return 0;
}
```

book.cpp

```cpp
Book::Book(int num, string title, string author, string genre, int page){
    this->num = num;
    this->title = title;
    this->author = author;
    this->genre = genre;
    this->page = page;

}

void Book::display() {
    cout << "Book Number: " << num << endl;
    cout << "Title: " << title << endl;
    cout << "Author: " << author << endl;
    cout << "Genre: " << genre << endl;
    cout << "Pages: " << page << endl <<endl;
}
```

Book.h

```cpp
class Book{
private:
    int num;
    string title;
    string author;
    string genre;
    int page;
public:
    Book(int num, string title, string author, string genre, int page);

    void display();
};
```

Output:

```
constructing 5 book objects
Book Number: 1
Title: b1
Author: b1a
Genre: b1g
Pages: 1

Book Number: 2
Title: b2
Author: b2a
Genre: b2g
Pages: 10

Book Number: 3
Title: b3
Author: b3a
Genre: b3g
Pages: 11

Book Number: 4
Title: b4
Author: b4a
Genre: b4g
Pages: 100

Book Number: 5
Title: b5
Author: b5a
Genre: b5g
Pages: 101

Book Number: 1
Title: b1
Author: b1a
Genre: b1g
Pages: 1
```

Problem 5:

Main.cpp

```cpp
int main(){
    WaterLevelMonitor monitor;
    cout << "Current water level:" << monitor.getCurrentLevel() << endl;
    cout << "Average water level:" << monitor.getAverageLevel() << endl;
    cout << "Highest water level:" << monitor.getHighestLevel() << endl;
    cout << "Lowest water level:" << monitor.getLowestLevel() << endl;

    double level;

    for (int i = 0; i<10; i++)
    {
        cout << "input the " <<i+1<< "th data: ";
        cin >> level;
        monitor.receiveData(level);
        cout << "Current water level:" << monitor.getCurrentLevel() << endl;
        cout << "Average water level:" << monitor.getAverageLevel() << endl;
        cout << "Highest water level:" << monitor.getHighestLevel() << endl;
        cout << "Lowest water level:" << monitor.getLowestLevel() << endl <<endl;
        cout << "count level: "<< monitor.getCountLevel() << endl;
        cout << "sum level: "<< monitor.getSumLevel() << endl;
    }
    system("pause");
    return 0;
}
```

WaterLevelMonitor.cpp

```cpp
WaterLevelMonitor::WaterLevelMonitor() {
    currentLevel = 0.0;
    averageLevel = 0.0;
    highestLevel = 0.0;
    lowestLevel = -1.0;
    countLevel = 0;
    sumLevel = 0.0;
}
void WaterLevelMonitor::receiveData(double level) {
    currentLevel = level;
    countLevel++;
    if (level > highestLevel) {
        highestLevel = level;
    }

    if (level < lowestLevel) {
        lowestLevel = level;
    }
    if (lowestLevel == -1.0)
        lowestLevel = level;
    sumLevel += level;
    averageLevel = sumLevel / countLevel;
    // cout << "count level: "<< countLevel << endl;
    // cout << "sum level: "<< sumLevel << endl;
}
double WaterLevelMonitor::getCurrentLevel() const {
    return currentLevel;
}
double WaterLevelMonitor::getAverageLevel() const {
    return averageLevel;
}
double WaterLevelMonitor::getHighestLevel() const {
    return highestLevel;
}
double WaterLevelMonitor::getLowestLevel() const {
    return lowestLevel;
}
double WaterLevelMonitor::getCountLevel(){
    return countLevel;
}
double WaterLevelMonitor::getSumLevel(){
    return sumLevel;
}
```

WaterLevelMonitor.h

```
1   class WaterLevelMonitor
2   {
3   private:
4       double currentLevel;
5       double averageLevel;
6       double highestLevel;
7       double lowestLevel;
8       int countLevel;
9       double sumLevel;
10  public:
11      WaterLevelMonitor();
12
13      void receiveData(double level);
14      double getCurrentLevel() const;
15      double getAverageLevel() const;
16      double getHighestLevel() const;
17      double getLowestLevel() const;
18      double getCountLevel();
19      double getSumLevel();
20  };
```

Output:

```
Current water level:0
Average water level:0
Highest water level:0
Lowest water level:-1
input the 1th data: 100
Current water level:100
Average water level:100
Highest water level:100
Lowest water level:100

input the 2th data: 2
Current water level:2
Average water level:51
Highest water level:100
Lowest water level:2

input the 3th data: 5
Current water level:5
Average water level:35.6667
Highest water level:100
Lowest water level:2

input the 4th data: 7
Current water level:7
Average water level:28.5
Highest water level:100
Lowest water level:2

input the 5th data: 1
Current water level:1
Average water level:23
Highest water level:100
Lowest water level:1

input the 6th data: 2
Current water level:2
Average water level:19.5
Highest water level:100
Lowest water level:1

input the 7th data: 3
Current water level:3
Average water level:17.1429
Highest water level:100
Lowest water level:1

input the 8th data: 5
Current water level:5
Average water level:15.625
Highest water level:100
Lowest water level:1

input the 9th data: 7
Current water level:7
Average water level:14.6667
Highest water level:100
Lowest water level:1
```

```
input the 10th data: 8
Current water level:8
Average water level:14
Highest water level:100
Lowest water level:1
```