Experiment No. 07 and 08

For all of the exercises below save your code on the learning management system (LMS) and give the screen shot of the output you get on the console in the space provided after every exercise.

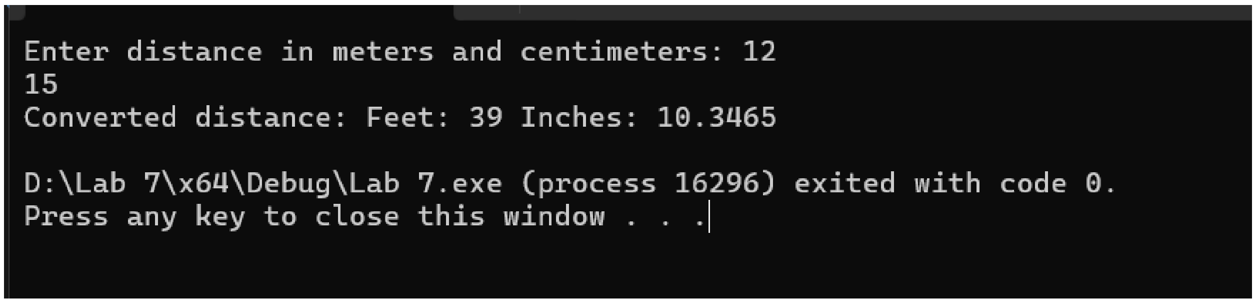
Exercise 1: (15 points)

Create two classes **DM** and **DB** that store the value of distances. DM stores distance in *meters* and *centimeters* and DB converts them to *feet* and *inches*.

Write a program that can read values for the class objects and converts them. Use a *friend* function to carry out the conversion operation by getting values from the user. The object that stores the results should be a DB object, depending on the units in which the results are required. The display should be in the format of feet and inches.

```cpp
#include <iostream>
using namespace std;

class DB;

class DM {
private:
    int meters;
    int centimeters;

public:

    DM();
    DM(int m, int cm);

    friend void convertToDB(const DM& dm, DB& db);
};

class DB {
private:
    int feet;
    float inches;

public:

    DB();
    DB(int ft, float in);


    void display() const {
        cout << "Feet: " << feet << " Inches: " << inches << endl;
    }

    friend void convertToDB(const DM& dm, DB& db);
};

DM::DM() {
    meters = 0;
    centimeters = 0;
}
DM::DM(int m, int cm) {
    meters = m;
    centimeters = cm;
}

DB::DB() {
    feet = 0;
    inches = 0;
}
DB::DB(int ft, float in) {
    feet = ft;
    inches = in;
}

void convertToDB(const DM& dm, DB& db) {

    float totalCentimeters = dm.meters * 100 + dm.centimeters;
    float totalInches = totalCentimeters / 2.54;
    db.feet = totalInches / 12;
    db.inches = totalInches - (db.feet * 12);
}

int main() {
    int meters, centimeters;
    cout << "Enter distance in meters and centimeters: ";
    cin >> meters >> centimeters;

    // Creating DM object and DB object for conversion
    DM dm(meters, centimeters);
    DB db;
    convertToDB(dm, db);
    cout << "Converted distance: ";
    db.display();

    return 0;
}
```

OUTPUT:



```
Enter distance in meters and centimeters: 12
15
Converted distance: Feet: 39 Inches: 10.3465

D:\Lab 7\x64\Debug\Lab 7.exe (process 16296) exited with code 0.
Press any key to close this window . . .
```

Exercise 2: (15 points)

Every Circle has a center and a radius. Create a class **CircleType** that can store the center, the radius, and the color of the circle. Since the center of a circle is a point in the x-y plane, create a class **PointType** to store the x and y coordinate. Use class *PointType* to define the class *CircleType*.

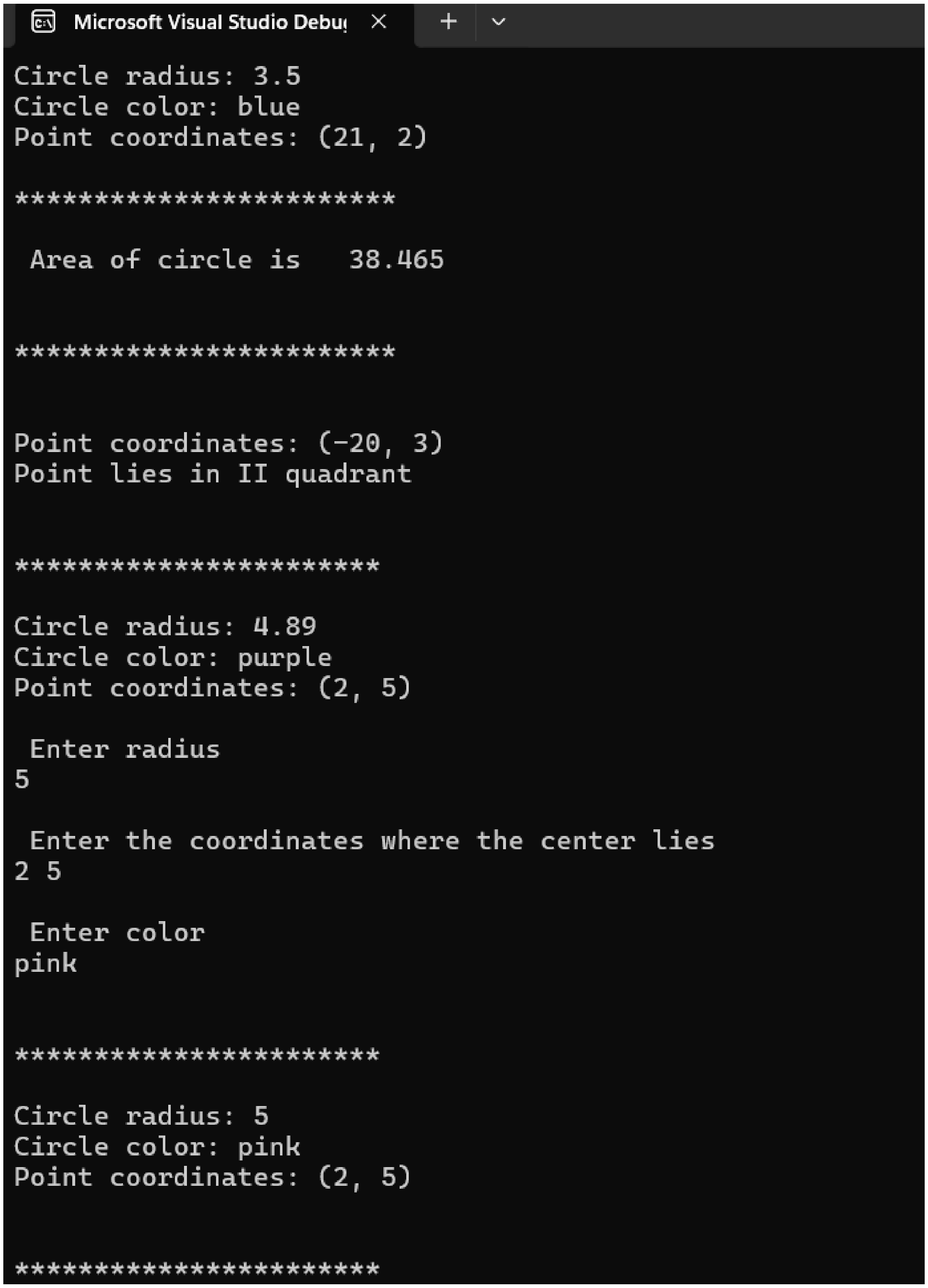
Provide *constructors* that enable objects of these classes to be initialized when they are declared. The constructors should contain default values in case no initializes are provided.
The definition of class *CircleType* and class *PointType* is as under: (you may define additional functions if you require any).

```
#include <iostream>
#include <cmath>
#include <string>
using namespace std;

class PointType {
    int x;
    int y;

public:

    PointType() { x = 0; y = 0; }
```

```cpp
    PointType(int x_val, int y_val) {
        x = x_val;
        y = y_val;
    }

    int getX() const { return x; }
    int getY() const { return y; }

    void setX(int x_val) { x = x_val; }
    void setY(int y_val) { y = y_val; }


    void print() const {
        cout << "Point coordinates: (" << x << ", " << y << ")" << endl;
    }

    int checkquad() const {
        if (x == 0 && y == 0)
            return 0;
        else if (x > 0 && y > 0)
            return 1;
        else if (x < 0 && y > 0)
            return 2;
        else if (x < 0 && y < 0)
            return 3;
        else if (x > 0 && y < 0)
            return 4;
        else
            return -1;
    }
};

class CircleType {
    double radius;
    string color;
    PointType center;

public:

    CircleType() { radius = 0; color = "black"; }

    CircleType(int x, int y, double r, const string& c) {
        radius = r;
        color = c;
        center = PointType(x, y);
    }

    // Print function
    void print() const {
        cout << "Circle radius: " << radius << endl;
        cout << "Circle color: " << color << endl;
        center.print();
    }

    double calc_area() const {
        return 3.14 * radius * radius;
    }

    double calc_circumference() const {
        return 2 * 3.14 * radius;
    }
    void setparam(int x, int y, double r, const string& c) {
        center = PointType(x, y);
        radius = r;
        color = c;
    }
};

int main() {
    CircleType C(21, 2, 3.5, "blue");
    cout << "\n************************\n" << endl;
    C.print();
    cout << "\n************************\n" << endl;
    cout << " Area of circle is    " << C.calc_area() << endl;
    cout << "\n\n************************\n\n" << endl;

    PointType P(-20, 3);
    int p = P.checkquad();
    P.print();

    switch (p) {
    case 0:
        cout << "Point lies at center" << endl;
        break;
    case 1:
        cout << "Point lies in I quadrant" << endl;
        break;
    case 2:
        cout << "Point lies in II quadrant" << endl;
```

```cpp
            break;
        case 3:
            cout << "Point lies in III quadrant" << endl;
            break;
        case 4:
            cout << "Point lies in IV quadrant" << endl;
            break;
        default:
            cout << "INVALID";
            break;
        }

        double r;
        int x, y;
        std::string col;

        CircleType circ(2, 5, 4.89, "purple");
        cout << "\n\n***********************\n\n";
        circ.print();

        cout << "\n Enter radius \n";
        cin >> r;
        cout << "\n Enter the coordinates where the center lies \n";
        cin >> x >> y;
        cout << "\n Enter color \n";
        cin >> col;

        circ.setparam(x, y, r, col);
        std::cout << "\n\n***********************\n\n";
        circ.print();
        std::cout << "\n\n***********************\n\n";
        return 0;
}
```

Output:



Microsoft Visual Studio Debu

```
Circle radius: 3.5
Circle color: blue
Point coordinates: (21, 2)

************************

 Area of circle is   38.465


************************


Point coordinates: (-20, 3)
Point lies in II quadrant


***********************

Circle radius: 4.89
Circle color: purple
Point coordinates: (2, 5)

 Enter radius
5

 Enter the coordinates where the center lies
2 5

 Enter color
pink


***********************

Circle radius: 5
Circle color: pink
Point coordinates: (2, 5)


***********************
```