University American College Skopje

Course: Object Programming

Classes and Objects

Exercises

Prepared by: Biljana Stojčevska, M.Sc.
Edited for 2017/18 by Ognen Spiroski, MSc

Exercise 1

Write a program that contains a class **my_class** with a private integer member **a** and function members that access **a**.

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

the class

```
//Definition of my_class
class my_class {
   int a;//data member a
public:
   void set_a(int b);
  //declaration for set _a
   int get_a();
  //declaration for get_a
};
```

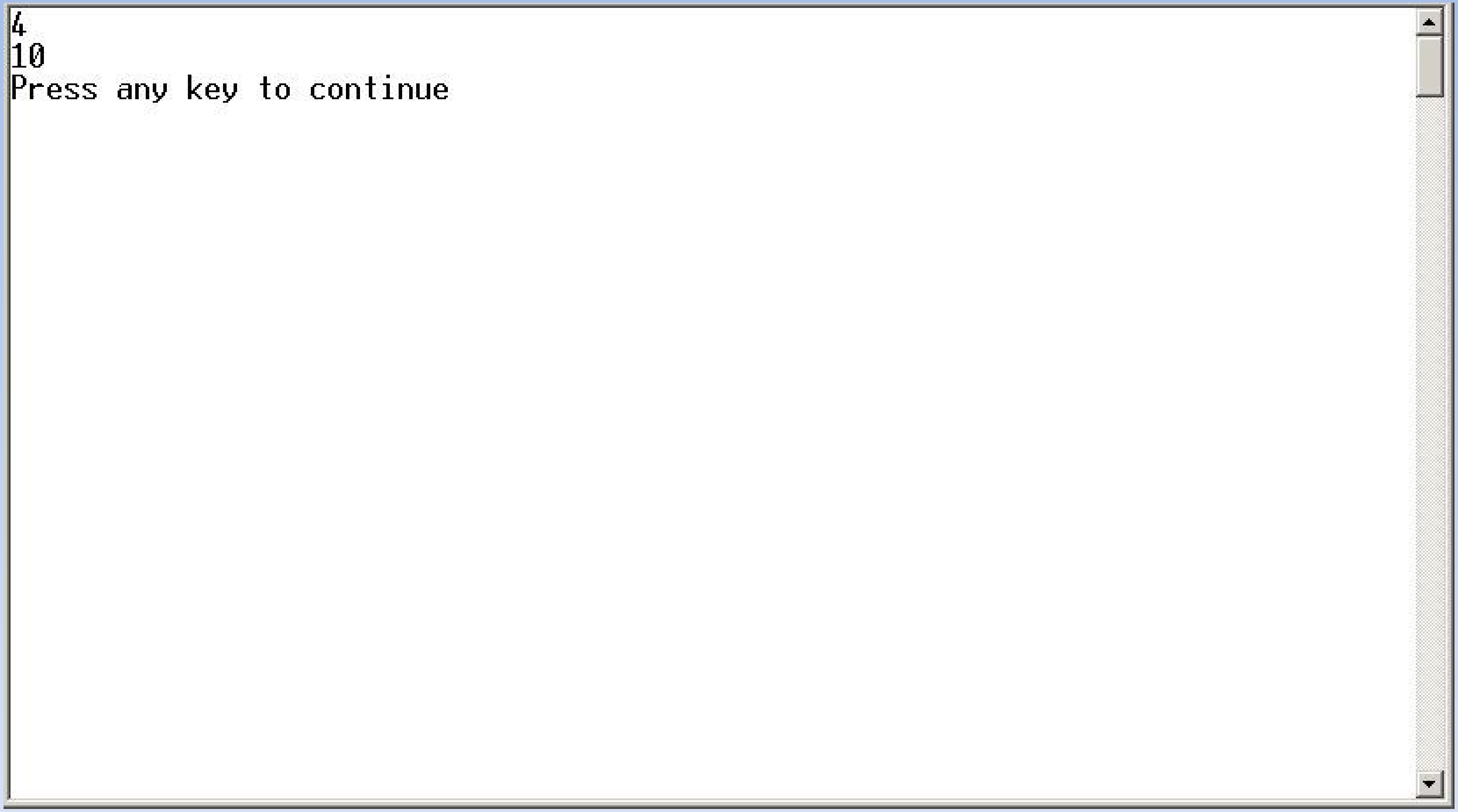
```
//main() function
int main(){

   my_class ob1, ob2;
   //create two objects of my_class

   ob1. set _a(4);
   ob2. set _a(10);
   cout<<ob1.get_a()<<endl;
   cout<<ob2.get_a()<<endl;

}
```

the main() function

```
//definition for set_a
void my_class:: set _a(int b){
   a=b;
}

// definition for get_a
int my_class:: get_a(){
   return a;
}
```

the function member definitions

Output for Exercise 1



```
4
10
Press any key to continue
```

Exercise 2

Write a program with a class that represents a rectangle with decimal values of its dimensions. The class should have function members for access to the data members, and function members that return the perimeter and the area of the rectangle.

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

the class

```
//definition of the class rectangle
class rectangle {
    double a, b;//data members
public:
    void set_a(double
    broj);//declaration of set_a
    void set_b(double broj);//
    declaration of set_b

    double get_a();//declaration of
    get_a
    double get_b();//declaration of
    get_b

    double perimeter();//declaration
    of perimetar
    double area();//declaration of
    plostina

};
```

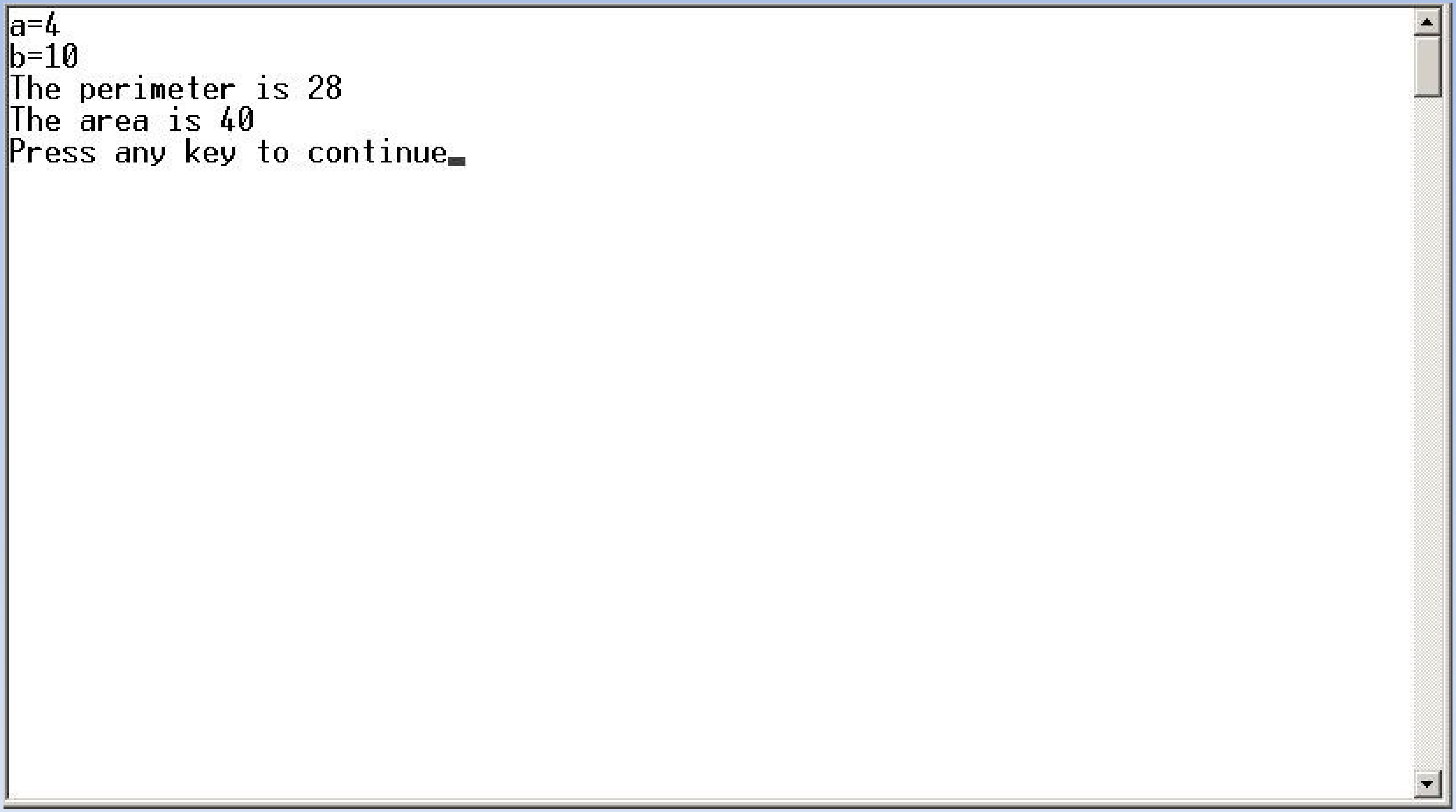
```
// definition of set_a
void rectangle:: set_a(double number){
    a= number;
}

// definition of set_b
void rectangle:: set_b(double number){
    b= number;
}

// definition of get_a
double rectangle:: get_a(){
    return a;
}

// definition of get_b
double rectangle:: get_b(){
    return b;
}

// definition of perimeter
double rectangle::perimeter(){
    return 2*a+2*b;
}

// definition of area
double rectangle::area(){
    return a*b;
}
```

the function member definitions

```
int main(){

    rectangle P;//create a rectangle
    object

    P.set_a(4);
    P.set_b(10);

    cout<<"a="<<P.get_a()<<endl;
    cout<<"b="<<P.get_b()<<endl;

    cout<<"The perimeter is
    "<<P.perimeter()<<endl;
    cout<<"The area is
    "<<P.area()<<endl;

}
```

the main() function

Output for Exercise 2



```
a=4
b=10
The perimeter is 28
The area is 40
Press any key to continue
```

Exercise 3

A program for the class **Pixel**. A pixel represents a point that has two integer coordinates, x and y.

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

the class

```
class Pixel {
public:
   int getx() { return x; }
   int gety() { return y; }
   void set ();
   void DrawPixel();
   void Copy(Pixel &);
private:
   int x, y;
};
```

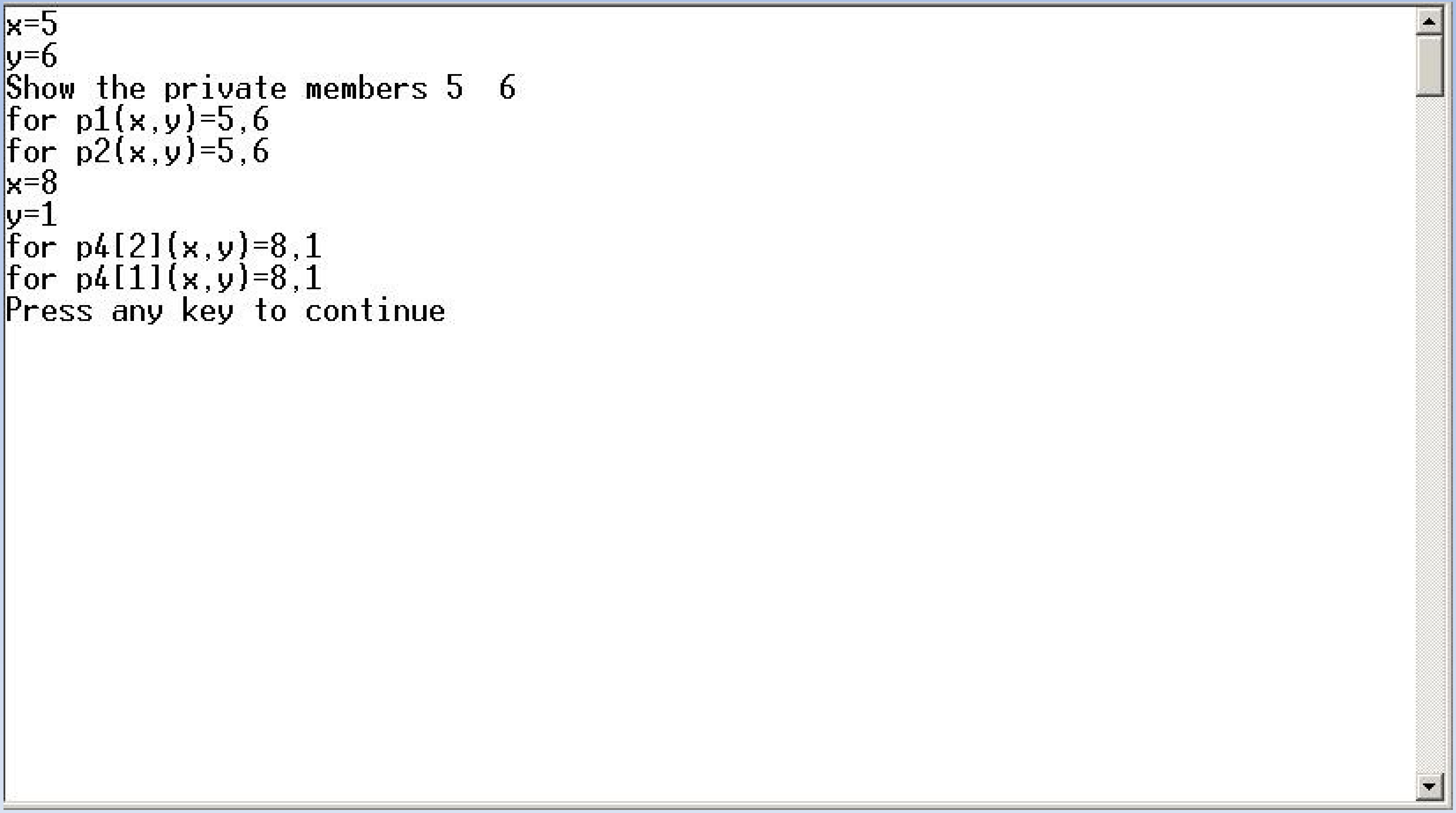
```
void Pixel::set () {
   cout<<"x=";   cin>>x;
   cout<<"y=";   cin>>y;
}

void Pixel::DrawPixel() {
   cout<<"(x,y)="<<x<<","<<y<<endl;
}

void Pixel::Copy(Pixel & Source) {
   x=Source.x;
   y= Source.y;
}
```

the function member definitions

```
int main()
{
   Pixel p1,p2,p4[5];
   p1.set();
   cout << "Show the private members ";
   cout <<p1.getx()<<"  "<<p1.gety()<<endl;
   cout << "for p1";
   p1.DrawPixel();
   p2.Copy(p1);

   cout << "for p2";
   p2.DrawPixel();
   p4[2].set();

   cout << "for p4[2]";
   p4[2].DrawPixel();

   p4[1].Copy(p4[2]);

   cout << "for p4[1]";
   p4[1].DrawPixel();


 }
```

the main() function

Output for Exercise 3



```
x=5
y=6
Show the private members 5  6
for p1(x,y)=5,6
for p2(x,y)=5,6
x=8
y=1
for p4[2](x,y)=8,1
for p4[1](x,y)=8,1
Press any key to continue
```

Exercise 4

Create a class **Square**. It should contain a private double member area, and corresponding get() and set() functions. It should also contain a function that returns the vale of the diagonal of the square (a double value).

If a is the area of the square and d is the diagonal of the square, then $d = \sqrt{2 * a}$

To obtain the use of the square root function, perform the following steps:

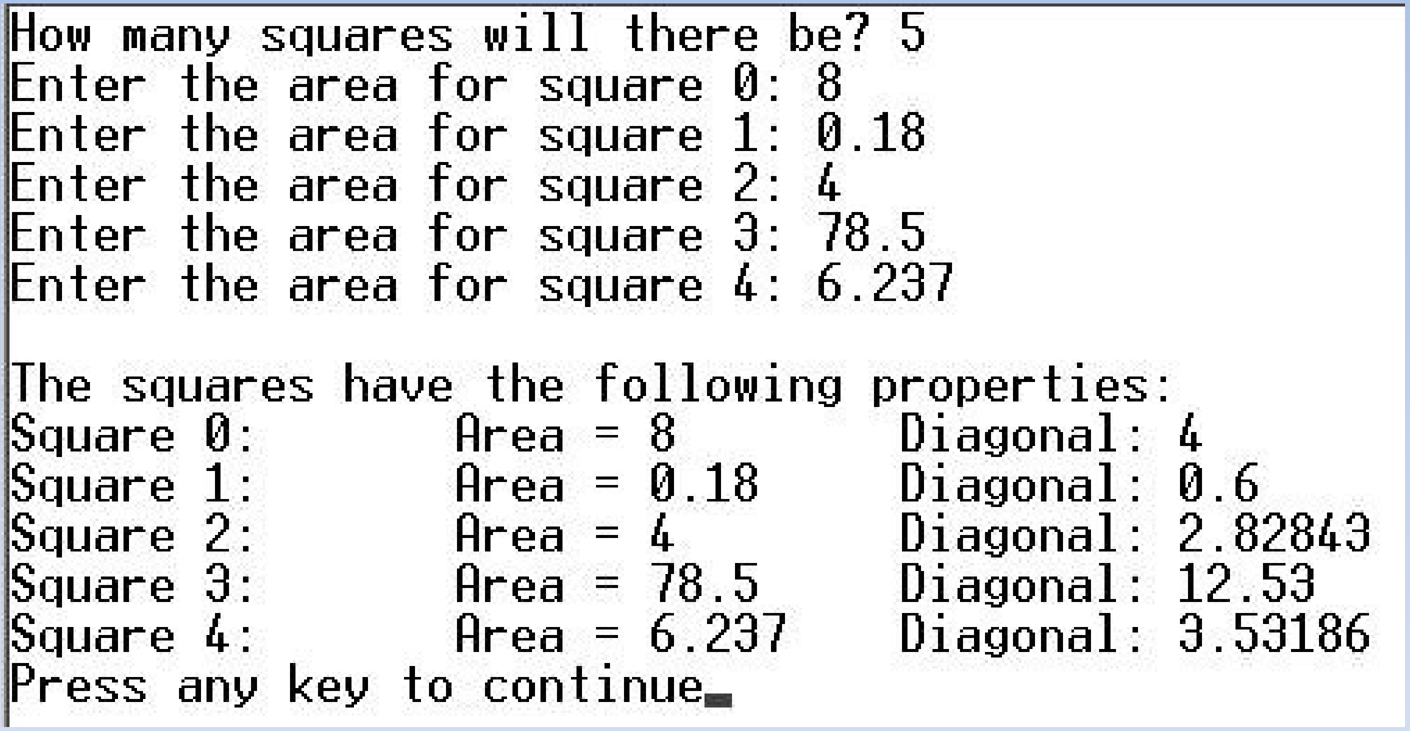
1. #include <cmath> at the beginning of your program
2. The square root function is now available as the sqrt() function (it returns a double value)

Exercise 4

In the main() program, create an array of Square objects (at most 100). Then input the number of squares you wish to define through the keyboard. Then enter the areas for those squares through the keyboard.

After you enter the areas of all the squares, output their corresponding areas and calculated diagonals.

An Example Output for Exercise 4



```
How many squares will there be? 5
Enter the area for square 0: 8
Enter the area for square 1: 0.18
Enter the area for square 2: 4
Enter the area for square 3: 78.5
Enter the area for square 4: 6.237

The squares have the following properties:
Square 0:       Area = 8        Diagonal: 4
Square 1:       Area = 0.18     Diagonal: 0.6
Square 2:       Area = 4        Diagonal: 2.82843
Square 3:       Area = 78.5     Diagonal: 12.53
Square 4:       Area = 6.237    Diagonal: 3.53186
Press any key to continue_
```

Exercise 5

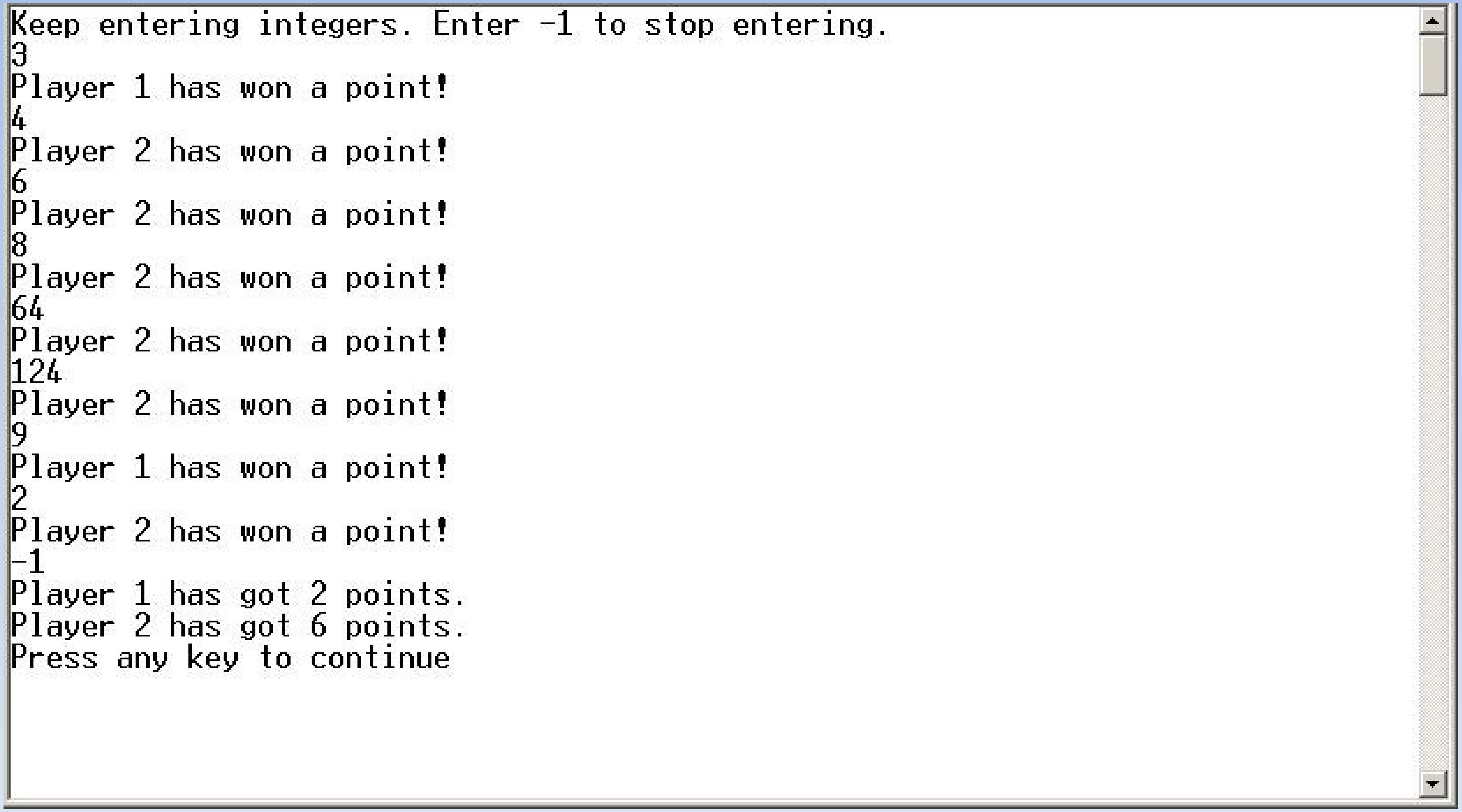
Create a class **Player**. It should contain a private int member point, and corresponding get() and set() functions.

In the main() program, define two Player objects and set their points to zero. Then keep entering integers until you enter -1.

When an odd integer is entered, Player 1 should increase its points by 1. When an even integer is entered, Player 2 should increase its points by 1. A message about which player has increased its points should appear immediately.

After -1 is entered, the total number of points should appear on the screen for both players.

An Example Output for Exercise 5



```
Keep entering integers. Enter -1 to stop entering.
3
Player 1 has won a point!
4
Player 2 has won a point!
6
Player 2 has won a point!
8
Player 2 has won a point!
64
Player 2 has won a point!
124
Player 2 has won a point!
9
Player 1 has won a point!
2
Player 2 has won a point!
-1
Player 1 has got 2 points.
Player 2 has got 6 points.
Press any key to continue
```

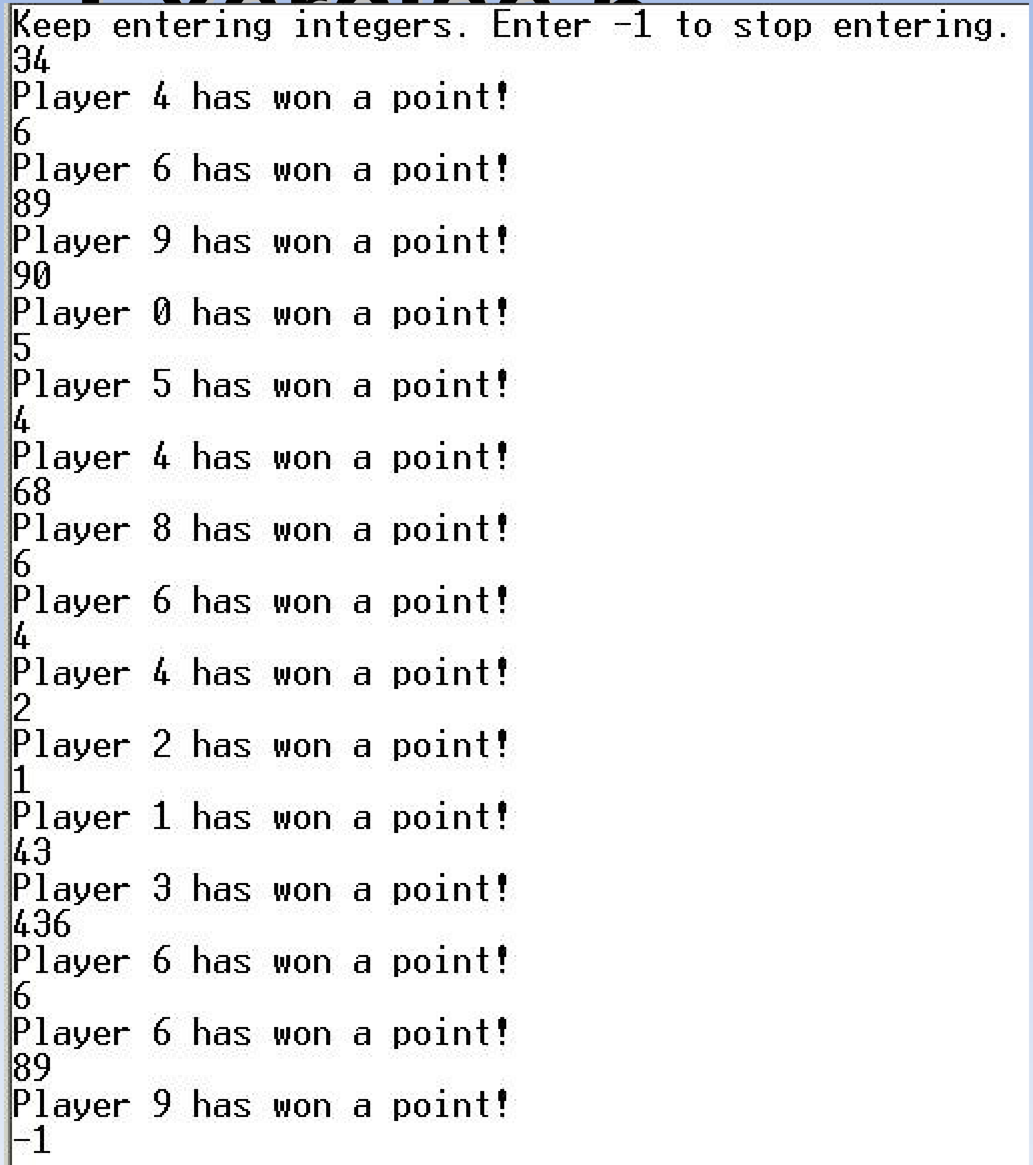
Exercise 6

Modify Exercise 4 by creating an array of 10 Player objects.

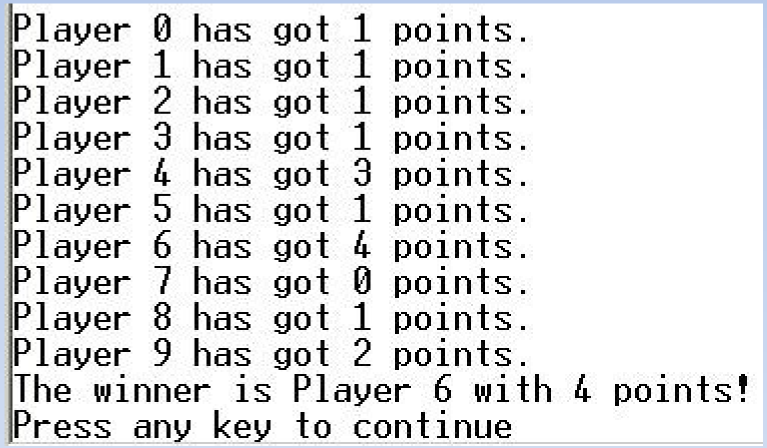
Then, keep entering integers until you enter -1. After an integer is entered, obtain the last digit of the integer and increase the points by one of the player with the index value in the array as the value of the digit. Display a message which player increased their points.

After -1 has been entered, display the obtained points for all players and declare the winner (i.e. the player that has got the most points).

An Example Output for Exercise 6



```
Keep entering integers. Enter -1 to stop entering.
34
Player 4 has won a point!
6
Player 6 has won a point!
89
Player 9 has won a point!
90
Player 0 has won a point!
5
Player 5 has won a point!
4
Player 4 has won a point!
68
Player 8 has won a point!
6
Player 6 has won a point!
4
Player 4 has won a point!
2
Player 2 has won a point!
1
Player 1 has won a point!
43
Player 3 has won a point!
436
Player 6 has won a point!
6
Player 6 has won a point!
89
Player 9 has won a point!
-1
```



```
Player 0 has got 1 points.
Player 1 has got 1 points.
Player 2 has got 1 points.
Player 3 has got 1 points.
Player 4 has got 3 points.
Player 5 has got 1 points.
Player 6 has got 4 points.
Player 7 has got 0 points.
Player 8 has got 1 points.
Player 9 has got 2 points.
The winner is Player 6 with 4 points!
Press any key to continue
```