Problem 35 marks

Particle collision is the fundamental principle behind the invention of cathode ray tube (CRT) televisions (TVs) and computer monitors. The CRT takes particles, called electrons, from the cathode, speeds them up using electromagnets and then smashes them into phosphor particles on the screen. The collision between the electron and phosphor particles results in a lighted spot, or pixel on your TV or computer monitor.

Develop a program to be used as a **particle collision detector**. This program determines whether two particles are colliding. For simplicity, particles are assumed to be sphere in shape. Thus, each particle can be represented by its center point and radius.

Your program should provide a class named ***Particle***, for representing a particle. The class has the following members:

a. Private data:
   i. The coordinates of center point, ***x***,***y*** and ***z***, respectively.
   ii. The radius, ***r***

b. Public operations:
   i. A default constructor.
   ii. A mutator function named ***read***. This function reads input from the keyboard to set the attributes ***x***, ***y***, ***z*** and *r*.
   iii. A function named **print** that displays all the attributes onto the screen.
   iv. An operator **-** that will be used for operations like:

particle1 – particle2

where ***particle1*** and ***particle2*** are of type **Particle** . The above operation results in the distance between those two particles.

The distance between two particles is calculated by:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

where (x_1, y_1, z_1) and (x_2, y_2, z_2) are the coordinates of the center points for the two particles respectively.

Your program should also define a stand alone or ordinary function named ***collision*** to test whether two particles are colliding. A collision is detected when the distance between two particles is less than or equal to the sum of their radius. The following figures illustrate this scenario.

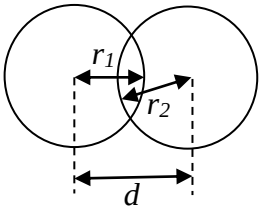

Figure 1: Collision

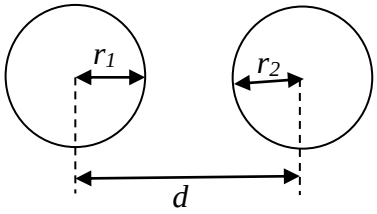

Figure 2: No collision

where r_1 and r_2 are the radius of two particles respectively, and ***d*** is the distance between the particles from their centers. A collision is detected in Figure 1 as $d \leq (r_1 + r_2)$, while there is no collision in Figure 2 as $d \geq (r_1 + r_2)$.

The function ***collision*** should receive two parameters of type ***Particle*** representing the two particles to be tested for collision. The function returns **true** if a collision is detected, otherwise **false**. Furthermore, specify the function to be a friend to the class ***Particle***. Also, the function should use the operator – from the class ***Particle*** to determine the distance between two particles.

Finally, utilize the class ***Particle*** and the function ***collision*** into the main function in order to realize the particle collision detector.

Your program should look like as the example runs illustrated in Figure 3. The program should prompt an appropriate message whether a collision is detected or not. If a collision is detected, beside the prompt, the program should also display the information of the two particles.

As for the assessment criteria, refer to Table 1.

```
Enter data for the first particle:
Center: 0 0 0
Radius: 2

Enter data for the second particle:
Center: 2 2 2
Radius: 3

The particles are colliding.

First particle:
Center: (0,0,0)
Radius: 2

Second particle:
Center: (2,2,2)
Radius: 3
```

(a)

```
Enter data for the first particle:
Center: 0 0 0
Radius: 2

Enter data for the second particle:
Center: 10 20 30
Radius: 3

No collision detected.
```

(b)

Figure 3: Example runs of the program with (a) a collision is detected and (b) no collision detected. Note that the bold texts are user input.

Table 1: Assessment Criteria

No	Criteria	Marks
	Program structure:	
P1	The program is able to run	5
P2	Proper styles, *e.g.* indentation and comments have been applied	1
P3	Use an appropriate structure for the program (*e.g.* all required header files are included, the function `main` is properly written, *etc.*)	2
	Class declaration:	
C1	Declaration of class **Particle** (overall)	2
C2	Declaration all the member variables of the class	1
C3	The default constructor	2
C4	The method **read**	3
C5	The method **print**	2
C6	The **operator -**	3
	Stand alone function:	
F1	Declaration of the function **collision**	1
F2	Specifying the function **collision** as friend to the class **Particle**	1
F3	Definition code for the function **collision** (overall)	1
F4	Code for the function **collision:** determining the distance	1
F5	Code for the function **collision:** determining collision	2
	The main function:	
M1	Creating two particles.	1
M2	Entering data for each particle.	2
M3	Collision testing	1
M4	Displaying a proper prompt whether a collision detected or not	2
M5	Displaying particles' information when a collision is detected.	2
	Total	**35**