```
1: ////////////////////////////////////////////////////////////////
   ///////////////
2: // Faculty of Computing, Universiti Teknologi Malaysia
3: // SCSJ1023- Programming Technique II
4: // Semester 2, 2017/2018
5: // Mid Term Test, Part B (Long Question)
6: // SOLUTIONS
7: //
8: // Prepared by: Jumail Bin Taliba (jumail@utm.my)
9: // 15 March 2018
10: ////////////////////////////////////////////////////////////////
   ///////////////
11:
12: #include<iostream>
13: #include<cmath>
14:
15: using namespace std;
16:
17: class Particle;
18:
19: bool collision(Particle, Particle);
20:
21: class Particle{    // 1m class declaration
22:       private:
23:               int x,y,z,r;     // 1m - declaration of attributes
24:
25:       public:
26:             Particle(){       // 0.5m - constructor header
27:                 x=y=z=r=0;    // 1m   - body
28:             }
29:
30:             void print() const   // 0.5m - method header
31:             {
32:                cout << "Center: (" << x <<"," << y << "," << z
   << ")     "
33:                     << "Radius: " << r << endl;   // 1m - method
   body
34:             }
35:
36:             double operator-(Particle right)   // 1m - header
37:             {
38:                int dx = x - right.x;   // 0.5m
39:                int dy = y - right.y;   // 0.5m
```

```
40:                double d = sqrt(dx*dx + dy*dy);  // 1m
41:
42:                return d;  //  0.5m
43:             }
44:
45:             bool operator>(Particle right)  // 1m - header
46:             {
47:                bool result = r > right.r;  // 1m
48:                return result;   // 0.5m
49:             }
50:
51:
52:             void read()  // 0.5m - header
53:             {
54:                   // 0.5m variable declarations
55:                   string attrib;
56:                   string var;
57:                   string sValue;
58:                   int value;
59:                   int len;
60:
61:                   cout << "Attribute and value=> ";
62:
63:                   while ( getline(cin,attrib) && (attrib!="")){
   // 2m,   loop structure.  while=1m, getline=1m, second condition
64:
65:                     len = attrib.length();  // 0.5m
66:
67:                     var = attrib.substr(0,1);       // 1m -
   extract attribute name
68:                     sValue= attrib.substr(2,len-2); // 1m -
   extract attribute value
69:                     value = atoi(sValue.c_str());   // 1m -
   convert to number
70:
71:                     // 4m. set attribute accordingly. Each: 0.5m
   condition, 0.5m set value.
72:                     if (var.compare("x")==0) x = value;
73:                     else if (var.compare("y")==0) y = value;
74:                     else if (var.compare("z")==0) z = value;
75:                     else r = value;
76:
```

```
77:                     cout << "Attribute and value=> ";
78:                 }
79:             }
80:
81:             friend bool collision(Particle, Particle);  // 1m - specify the function collision() to be a friend to the class
82: };
83:
84: bool collision(Particle particle1, Particle particle2)  // 1m - function header
85: {
86:      double d = particle1 - particle2;  // 2m  using the overloaded operator- to determine the distance
87:      int r = particle1.r + particle2.r; // 1m
88:      bool result = (d<r);               // 1m
89:      return result;                     // 0.5m
90: }
91:
92: int main()
93: {
94:     Particle p1, p2;  // 1m. Creating two Particle objects
95:     cout << "Particle 1: " << endl << endl;
96:     p1.read();        // 1m. Invoking the read method
97:
98:     cout << endl;
99:
100:    cout << "Particle 2: " << endl << endl;
101:    p2.read();  // 1m. Invoking the read method
102:
103:    cout << endl << endl;
104:
105:    if (collision(p1,p2))   // 2m testing for a collision
106:       cout << "** Collision detected **" << endl;   // 1m
107:    else cout << "** No collision **" << endl;  // 1m
108:
109:    cout << endl;
110:    cout << "Particle 1: " << endl;
111:    p1.print();  // 1m invoking the print method
112:
113:    cout << endl;
114:    cout << "Particle 2: " << endl;
115:    p2.print();  // 1m invoking the print method
116:
```

```
117:     cout << endl;
118:
119:     if (p1 > p2)  // 1m  using the overloaded operator for
     comparision
120:         cout << "Larger: Particle 1" << endl;  // 1m
121:     else
122:         cout << "Larger: Particle 2" << endl;  // 1m
123:
124:
125:
126:     return 0;
127: }
128:
```