Puzzle from Monday

```cpp
puzzle.cpp
Sphere *CreateUnitSphere() {
  Sphere s(1);
  return &s;
}
int main() {
  Sphere *s = CreateUnitSphere();
  someOtherFunction();
  double r = s->getRadius();
  double v = s->getVolume();
  return 0;
}
```

Takeaway:

**Heap Memory:**
As programmers, we can use heap memory in cases where the lifecycle of the variable exceeds the lifecycle of the function.

1. The only way to create heap memory is with the use of the `new` keyword. Using `new` will:
   - 
   - 
   - 

2. The only way to free heap memory is with the use of the `delete` keyword. Using `delete` will:
   - 
   - 

3. Memory is never automatically reclaimed, even if it goes out of scope. Any memory lost, but not freed, is considered to be “leaked memory”.

```cpp
heap1.cpp
int main() {
  int *p = new int;
  Sphere *s = new Sphere(10);
  return 0;
}
```

```cpp
heap2.cpp
int main() {
  Sphere *s1 = new Sphere();
  Sphere *s2 = s1;
  s2->setRadius(10);
  delete s2;
  delete s1;
  return 0;
}
```
Copying Memory – Deep Copy vs. Shallow Copy

Consider how each assignment operator changes the data:

<table>
<thead>
<tr>
<th>Type of LHS</th>
<th>Type of RHS</th>
<th>Data Changed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 8-9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>j</td>
<td>k</td>
</tr>
<tr>
<td>p</td>
<td>q</td>
<td>r</td>
</tr>
<tr>
<td>Line 11-12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>j</td>
<td>k</td>
</tr>
<tr>
<td>p</td>
<td>q</td>
<td>r</td>
</tr>
<tr>
<td>Line 14-15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>j</td>
<td>k</td>
</tr>
<tr>
<td>p</td>
<td>q</td>
<td>r</td>
</tr>
</tbody>
</table>

heap-puzzle1.cpp

5 int *x = new int;
6 int &y = *x;
7 y = 4;
8 cout << &x << endl;
9 cout << x << endl;
10 cout << *x << endl;
11 cout << &y << endl;
12 cout << y << endl;
13 cout << *y << endl;

heap-puzzle2.cpp

5 int *p, *q;
6 p = new int;
7 q = p;
8 *q = 8;
9 cout << *p << endl;
10 q = new int;
11 *q = 9;
12 cout << *p << endl;
13 cout << *q << endl;

heap-puzzle3.cpp

5 int *x;
6 int size = 3;
7 x = new int[size];
8 for (int i = 0; i < size; i++) {
9   x[i] = i + 3;
10 }
11 delete[] x;

joinSpheres.cpp

11 /*
12 * Creates a new sphere that contains the exact volume
13 * of the two input spheres.
14 */
15 Sphere joinSpheres(Sphere s1, Sphere s2) {
16   double totalVolume = s1.getVolume() + s2.getVolume();
17   double newRadius = std::pow((3.0 * totalVolume) / (4.0 * 3.141592654), 1.0/3.0);
18   Sphere result(newRadius);
19   return result;
20 }

CS 225 – Things To Be Doing:

1. Exam 0 is ongoing – ensure you’re signed up for it!
2. Finish up MP1 – Due Monday, Jan. 29 at 11:59pm
3. Complete lab_debug this week in lab sections (due Sunday)
4. POTDs are released daily, worth +1 extra credit point! 😊