You are going to analyze the complexity of several variations on some silly code. In each case, assume that \( n \) is properly declared and its value is read in from the user.

You should always give the strongest answer you can. For example, it’s better to say that code is \( O(\log n) \) than to say that it’s \( O(n) \) — as long as that’s true of course. :-)

1. int blah = 0;
   for (int p = 1; p <= n; p++) {
      for (int q = 1; q <= n; q++) {
         for (int r = 1; r <= n; r++) {
           blah = blah + r;
         }
      }
      blah = blah + p;
      System.out.println("Hello");
   }

   What is the big-oh time complexity of this code, in the worst case?

2. int blah = 0;
   for (int p = 1; p <= n; p++) {
      for (int q = 1; q <= n; q++) {
         for (int r = 1; r <= n; r++) {
            blah = blah + r;
            blah = blah + p;
            System.out.println("Hello");
         }
      }
   }

   What is the big-oh time complexity of this code, in the worst case?

3. int blah = 0;
   for (int p = 1; p <= n; p++) {
      for (int q = 1; q <= n; q++) {
         for (int r = 1; r <= 10000; r++) {
            blah = blah + r;
         }
      }
      blah = blah + p;
      System.out.println("Hello");
   }

   What is the big-oh time complexity of this code, in the worst case?
4. int blah = 0;
   for (int p = 1; p <= n; p++) {
       for (int q = 1; q <= n; q++) {
           for (int r = 1; r <= p; r++) {
               blah = blah + r;
           }
       }
       blah = blah + p;
       System.out.println("Hello");
   }

What is the big-oh time complexity of this code, in the worst case?

5. int blah = 0;
   for (int p = 1; p <= n; p++) {
       for (int q = 1; q <= n; q++) {
           for (int r = n; r <= 7n; r++) {
               blah = blah + r;
           }
       }
       blah = blah + p;
       System.out.println("Hello");
   }

What is the big-oh time complexity of this code, in the worst case?

6. Suppose that another variable, m, is also properly declared and its value is read in from the user.

   int blah = 0;
   int shrink = m;
   for (int p = 1; p <= n; p++) {
       while (shrink > 1) {
           shrink = shrink / 2;
           blah = blah + r;
       }
       blah = blah + p;
       System.out.println("Hello");
   }

What is the big-oh time complexity of this code, in the worst case?