I a) algorithm 1
First note that for loops are a pain to count. Much easier to convert them to while loops:

```java
public double f_alg1(double x, int k) {
    // Compute x^8
    double x8 = 1.0;
    for(int i = 0; i < 8; i++)
        x8 = x8*x;
    double s = 0.0;
    for(int n = 1; n <= k; n++) {
        double f = 1.0;
        for(int i = 1; i <= n; i++)// Compute n!
            f = f*i;
        s = s + x8/f; // Add next term
    }
    return s;
}
```

becomes:

```java
public double f_alg1(double x, int n) {
    double x8 = 1.0;
    int i=0;
    while( i < 8 )
        x8 = x8*x;  // 9 comparisons, 8 times through
        i++;
    double s = 0.0;
    int k = 1;
    while( k <= n ) {
        double f = 1.0;
        int i = 1;
        while( i <= k) // Compute n!
            f = f*i;
        s = s + x8/f; // Add next term
        n++;
    }
    return s;
}
```

the first 7 lines plus the return are: 1+1+9+8(2+2)+1+1+1 = 46 steps.

Next we notice that the number of times through the inner loop varies, so we know we'll break that portion out. First, we'll just count the outer loop, pretending the inner loop doesn't exist:

k+1+k(1+1+3+2)= 8k+1 steps.

Now, for the inner loop. The first pass is 2 comparisons, plus 1 time through the loop of 4 steps, or 2+4 steps. The second pass is 3 comparisons, plus 2 times through the loop of 4 steps, or 3+8 steps. The third pass is 4 comparisons, plus 3 times through the loop of 4 steps, or 4+12 steps, etc. The last pass is k+1 comparisons, plus k times through the loop of 4 steps, or k+1+4k steps. We now have a series: 2+4+3+8+4+12+...+k+1+4k. First, let's separate out the comparisons from the loop, leaving 2 series: 2+3+4+...+k+1 for the comparisons, and 4+8+12+...+4k for the loop contents.

If we take the first series and move the 1 to the front, we get 1+2+3+4+...+k, which we know is k(k+1)/2. We can the factor out a 4 from the second series to get 4(1+2+3+...+k), which we know is 4k(k+1)/2, or 2k(k+1). So the inner loop is k(k+1)/2 + 2k(k+1).

So we get a final tally of 46 + 8k+1 + k(k+1)/2 + 2k(k+1). Simplify: (5k^2)/2 + 21k/2 + 47, or just O(k^2).
I. a) algorithm 2
public double f_alg2(double x, int k) {
    // Compute x^8
    double x2 = x*x;
    double x4 = x2*x2;
    double x8 = x4*x4;
    double s = 0.0;
    double f = 1.0;
    for(int n = 1; n <= k; n++) {
        f = f*n; // Compute n!
        s = s + x8/f; // Add next term
    }
    return s;
}

Convert to whiles:
public double f_alg2(double x, int k) {
    double x2 = x*x;
    double x4 = x2*x2;
    double x8 = x4*x4;
    double s = 0.0;
    double f = 1.0;
    int n = 1;
    while(n <= k) {
        f = f*n; // Compute n!
        s = s + x8/f; // Add next term
        n++
    }
    return s;
}

With only 1 loop, this one is much easier. The first 6 lines are 9 steps. The loop has k+1 comparisons, k times through, and 7 steps inside, for a total of k+1+7k. And there is 1 step in the last line, resulting in, 8k+11, or O(k).

3) This matches the graph perfectly, the first being a quadratic curve and the second a linear one.