1. (16 pts) **Big-O**

For each of the functions *f(N)* given below, indicate the tightest bound possible (in other words, giving $O(2^N)$ as the answer to every question is not likely to result in many points). Unless otherwise specified, all logs are base 2. **You MUST choose your answer from the following** (not given in any particular order), each of which could be re-used (could be the answer for more than one of a) – h)):

$O(N^2)$, $O(N^{1/2})$, $O(N^{1/4})$, $O(\log^3 N)$, $O(N)$, $O(N^2 \log N)$, $O(N^5)$, $O(2^N)$, $O(N^3)$, $O(N^8)$, $O(\log^4 N)$, $O(N \log^3 N)$, $O(N^2 \log^2 N)$, $O(\log N)$, $O(1)$, $O(N^4)$, $O(N^N)$, $O(N^6)$, $O(N \log^2 N)$, $O(\log \log N)$

You do not need to explain your answer.

a) $f(N) = N^{1/2} + \log^3 N$ $O(N^{1/2})$

b) $f(N) = 100 \log N + N^{1/4}$ $O(N^{1/4})$

c) $f(N) = N^2 \log N^2 + 2 N \log^2 N$ $O(N^2 \log N)$

d) $f(N) = (N\cdot (100N + 5 + N^{3}))^2$ $O(N^8)$

e) $f(N) = 1000 \log \log N + \log N$ $O(\log N)$

f) $f(N) = \log_{16}(2^N)$ $O(N)$

g) $f(N) = N^2 \cdot (\log N^3 – \log N)+ N^{2}$ $O(N^2 \log N)$

h) $f(N) = (N \log N + 2N)^2$ $O(N^2 \log^2 N)$

2. (8 pts) **Big-Oh and Run Time Analysis:** Describe the worst case running time of the following pseudocode functions in Big-Oh notation in terms of the variable `n`. ***Showing your work is not required*** (although showing work **<u>may</u>** allow some partial credit in the case your answer is wrong – don’t spend a lot of time showing your work.). You MUST choose your answer from the following (not given in any particular order), each of which could be re-used (could be the answer for more than one of I. – IV.):

$O(n^2)$, $O(n^3 \log n)$, $O(n \log n)$, $O(n)$, $O(n^2 \log n)$, $O(n^5)$, $O(2^n)$, $O(n^3)$,
$O(\log n)$, $O(1)$, $O(n^4)$, $O(n^n)$, $O(n^6)$, $O(n^8)$, $O(n^7)$

```
  I. void sunny(int n, int x) {
         for (int k = 0; k < n; ++k)
            if (x < 50) {
               for (int i = 0; i < n; ++i)
                  for (int j = 0; j < i; ++j)
                      System.out.println(”x = ” + x);
            } else {
               System.out.println(”x = ” + x);
            }
     }
```

Runtime:

$O(n^3)$

```
 II. void warm(int n) {
         for (int i = 0; i < 2 * n; ++i) {
            j = 0;
            while (j < n) {
               System.out.println(”j = ” + j);
               j = j + 5;
            }
         }
     }
```

$O(n^2)$

```
III. int silly(int n, int m) {
         if (n < 1) return m;
         else if (n < 10)
           return silly(n/2, m);
         else
           return silly(n - 2, m);
     }
```

$O(n)$

```
 IV.
     void happy(int n) {
          for (int i = n*n; i > 0; i--) {
             for (int k = 0; k < n; ++k)
                System.out.println(”k = ” + k);
             for (int j = 0; j < i; ++j)
                System.out.println(”j = ” + j);
             for (int m = 0; m < 5000; ++m)
               System.out.println(”m = ” + m);
         }
       }
```

$O(n^4)$