Name: _____ **Roll Number:** _____

Ques. 1 Do as Directed: (5 Marks)

1. **When determining the efficiency of algorithm the time factor is measured by**
   (a) Counting microseconds
   (b) Counting the number of key operations
   (c) Counting the number of statements
   (d) Counting the kilobytes of algorithm

2. A machine took 200 sec to sort 200 names, using bubble sort. In 800 sec, it can approximately sort _____ names.

   (a) 400 names (b) 800 names (c) 750 names (d) 850 names

3. Which one of the following is false for the given recurrence relation?

$$T(n) = \begin{cases} 1, & n = 0, 1 \\ 2T(n/2), & n > 1 \end{cases}$$

   (a) $T(n) = O(n^2)$ (b) $T(n) = \theta(n\text{Log}n)$ (c) $T(n) = \Omega(n^2)$ (d) $T(n) = O(n\text{Log}n)$

4. Consider the following recurrence $T(n) = 3T(n/5) + \log n * \log n$

$$T(n) = \begin{cases} 1, & n = 0 \\ 3T(n/5) + logn * logn, & n > 0 \end{cases}$$

   What is the value of T(n)?

   (A) $\Theta(n^{\log_5 3})$
   (B) $\Theta(n^{\log_3 5})$
   (c) $\Theta(nLogn)$
   (D) $\Theta(Logn)$

5. For analysing an algorithm, which is better computing time?

   (a) O(100 logN) (b) O(N) (c) O(2N) (d) O(N logN) (e) $O(N^2)$

Ques. 2 Indicate the Running Time (Worst Case) for the following functions. (20 Marks)

1.
```
int f1(int n, int a[]) {
s = 0;
for (int i = 0; i < n; ++i) {
for (int j = 0; j < n; ++j) {
s = s + abs(a[i] - a[j]);
} }
return s;
}
```
$O(n^2)$

2.
```
int f2(int n, int a[]) {
s = 0;
for (int i = 0; i < n; ++i) {
if (a[i]>n)
for (int j = 0; j < i; ++j) {
s = s + a[i]*a[j];
} }
return s;
}
```
$O(n^2)$

3.
```
int f3(int n {
s = 0;
for (int i = 0; i < n; ++i) {
for (int j = 0; j < i*i; ++j) {
s = s + j;
} }
return s; }
```
$O(n^3)$

4.
```
void silly(int n) {
for (int i = 0; i < 1000; ++i) {
for (int j = 0; j < n; ++j) {
for (int k = 0; k < j; ++k)
System.out.println("k = " + k);
for (int m = 0; m < i; ++m)
System.out.println("m = " + m);
} } }
```
$O(n^2)$

5.
```
void silly(int n, int x, int y) {
for (int i = 0; i < n; ++i) {
if (x < y) {
for (int j = 0; j < n * n; ++j)
System.out.println("j = " + j);
} else
System.out.println("i = " + i);
} }
```
$O(n^3)$

6.
```
void silly(int n) {
for (int i = 0; i < n; ++i) {
j = 0;
while (j < n) {
System.out.println("j = " + j);
j++;
} } }
```
$O(n^2)$

7.
```
void silly(int n) {
for (int i = 0; i < n; ++i) {
for (int j = 0; j < i; ++j) {
System.out.println("j = " + j);
for (int k = 0; k < j; ++k)
System.out.println("k = " + k);
} } }
```
$O(n^3)$

8.
```
void silly(int n, int x, int y) {
for (int i = 0; i < n; ++i) {
if (x > y) {
for (int j = 0; j < n; ++j)
System.out.println("j = " + j);
for (int k = 0; k < n * n; ++k)
System.out.println("k = " + k);
} else
System.out.println("i = " + i);
} }
```
$O(n^3)$

9.
```
void silly(int n) {
for (int i = 0; i < n; i = i + 10) {
for (int j = 0; j < i; ++j) {
System.out.println("i = " + i);
System.out.println("j = " + j);
} } }
```
$O(n^2)$

10.
```
void silly (int n) {
for (int i = 0; i < n; ++i) {
for (int j = 0; j < i; ++j) {
System.out.println("j = " + j);
}
for (int k = 0; k < n * 3; ++k) {
System.out.println("k = " + k);
} } }
```
$O(n^2)$

11.
```
void silly(int n, int x, int y) {
for (int i = 0; i < n; ++i) {
if (x < y)
for (int k = 0; k < n * n; ++k){
System.out.println("k = " + k);
}
else
System.out.println("i = " + i);
} }
```
$O(n^3)$

12.
```
void silly(int n, int x, int y) {
for (int k = n; k > 0; k--)
if (x < y + n) {
for (int i = 0; i < n; ++i)
for (int j = 0; j < i; ++j)
System.out.println("y = " + y);
} else {
System.out.println("x = " + x);
} }
```
$O(n^3)$

13.
```
void silly(int n) {
for (int i = 0; i < n; ++i) {
for (int j = 0; j < n; ++j)
System.out.println("j = " + j);
for (int k = 0; k < i; ++k) {
System.out.println("k = " + k);
for (int m = 0; m < 100; ++m)
System.out.println("m = " + m);
} } }
```
$O(n^2)$

14.
```
int x=0, i;
    for(i = n; i >= 0; i--)
    {
    if((i % 3) == 0)
    break;
    else
    x += i;
    }
```
$O(1)$ or $O(3)$ or Constant Time

15.
```
sum = 0, j = -1;
    for(i = 0; i < n; i++)
        {
            if ( i > j)
                    sum = sum + 1;
            else
            {
                for(k = 0; k < n; k++)
                    sum = sum – 1;
            }
        }
```
$O(n)$

16.
```
void silly(int n) {
j = 0;
while (j < n) {
for (int i = 0; i < n; ++i) {
System.out.println("j = " + j);
}
j = j + 5;
} }
```
$O(n^2)$

17.
```
void function(int n)
        {
         int i = 1, s =1;
         while (s <= n)
         {      i++;
                s += i;
                printf("*");
         } }
```
$O(\sqrt{n})$

18.
```
for (int i = 0; i < n + 100; ++i) {
for (int j = 0; j < i * n ; ++j){
sum = sum + j;
}
for (int k = 0; k < n + n + n; ++k){
c[k] = c[k] + sum;
} }
```
$O(n^3)$

19.
```
for (int j = 4; j < n; j=j+2) {
val = 0;
for (int i = 0; i < j; ++i) {
val = val + i * j;
for (int k = 0; k < n; ++k){
val++;
} } }
```
$O(n^3)$

20.
```
for (int i = 0; i < n * 1000; ++i) {
sum = (sum * sum)/(n * i);
for (int j = 0; j < i; ++j) {
sum += j * i;
}
}
```
$O(n^2)$

Ques. 3 **For each of the functions below, give the simplest, big-O expression. For example, if the function where n + 20, then you should answer O(n):** **(10 Marks)**

1. $n^{1/3} + n^{1/4} + \log n$ $O(n^{1/3})$
2. $\log n^2 + \log^2 n$ $O((\log n)^2)$
3. $2^{3n} + 3^{2n}$ $O(9^n)$
4. $n! + 2^n$ $O(n!)$
5. $n^2 + 2^n$ $O(2^n)$
6. $(N + N + N + N)^2$ $O(N^2)$
7. $(N^2 + N) / N$ $O(N)$
8. $(N/2) \log (2N) + N$ $O(N \log N)$
9. $N \log N + N \log (N^2)$ $O(N \log N)$
10. $(N/3) \log (N^3) + 3N$ $O(N \log N)$

Ques. 4 **Write True/False:** **(5 Marks)**

1. If $f(n) = O(g(n))$ then $g(n) = \Omega(f(n))$. True
2. If $f(n) = O(g(n))$ then $f(n) = o(f(n))$. False
3. If $f(n) = O(g(n))$ and $g(n) = O(h(n))$ then $f(n) = O(h(n))$. True
4. If $f(n) = O(g(n))$ then for every n, $f(n) \leq g(n)$. False
5. If $f(n) = O(g(n))$ then $f(n) + g(n) = O(g(n))$. True