Name: ___Katie Prescott_________________ Date: __3/18/16__________

Use the Master Theorem to find the complexity of each recurrence relation listed below.

1. $T(n) = T\left(\frac{n}{2}\right) + n^2$

Complexity: $\boldsymbol{T(n) = \theta(n^2)}$

2. $T(n) = 4T\left(\frac{n}{2}\right) + n^2$

Complexity: $\boldsymbol{T(n) = \theta(n^2 \log_2 n)}$

3. $T(n) = 3T\left(\frac{n}{3}\right) + \sqrt{n}$

Complexity: $\boldsymbol{T(n) = \theta(n)}$

For each function below, write the recurrence relation for its running time and then use the Master Theorem to find its complexity.

4.
```
int f(int arr[], int n) {
     int sum = 0;

     for (int j = 0; j < n; ++j) {
          sum += arr[j];
     }

     return f(n / 2) + sum + f(n / 2);
}
```

Recurrence: $\boldsymbol{T(n) = 2T(n/2) + n}$

Complexity: $\boldsymbol{T(n) = \theta(n \log_2 n)}$

5.
```
void g(int n, int arrA[], int arrB[]) {
     for (int i = 0; i < n; ++i) {
          for (int j = 0; j < n; ++j) {
               arrB[j] += arrA[i];
          }
     }

     g(n / 2);
}
```

Recurrence: $\boldsymbol{T(n) = T(n/2) + n^2}$

Complexity: $\boldsymbol{T(n) = \theta(n^2)}$