P 44.

```
void fun (int n) {
    if (n < 2) return;
    else counter = 0;
    for (i = 1 to 8)
        fun (n/2);

    for (i = 1 to n^3)
        counter++;
}
```

We have, $T(n) = 8T\left(\frac{n}{2}\right) + n^3$

Here, $a = 8$, $b = 2$, $k = 3$, $p = 0$

Case 2. (a) $a = b^k$ and $p > -1$

$\therefore T(n) = \theta(n^{\log_b^a} \log^{p+1} n)$

$\Rightarrow T(n) = \theta(n^3 \log n)$

P 45.

```
temp = 1;
repeat          // n times
    for i = 1 to n  //
        temp++;
    n = n/2   // recursive call
until n <= 1
```

We have, $T(n) = T\left(\frac{n}{2}\right) + n$

$\therefore T(n) = O(n)$.

P 46.

```
fun (int n) {
    for (int i = 1; i <= n; i++)   // n times
        for (int j = 1; j <= n; j *= 2)   // log n times
            cout << "*";
}
```

$\therefore O(n \log n)$

P 47.

```
fun (int n) {
    for (int i = 1; i <= n/3; i++)   // n/3 times
        for (int j = 1; j <= n; j += 4)   // n/4 times
            cout << "*";
```

$\therefore O(n^2)$.

P48.

```
void function (int n) {
        if (n <= 1) return;
        if (n > 1) {

            cout << "*";
            function (n/2);

            function (n/2);
        }
}
```

We have, $T(n) = 2T\left(\frac{n}{2}\right) + 1$

Here, $a = 2$, $b = 2$, $k = 1$, $p = 0$

Case 2. (a). $a = b^k$, $p > -1$

$\therefore$ $T(n) = \Theta(n^{\log_b^a} \log^{p+1} n)$

$\Rightarrow$ $T(n) = \Theta(n)$

P 49.

```
fun(int n) {
    int i = 1;
    while (i < n) {
        int j = n;
        while (j > 0)
            j = j/2; // log n
        i = 2*i; // log n
    }
}
```

$\therefore$ $O(\log^2 n)$