# Advanced Time Complexity Worksheet

1.
```
for (int i = 1; i < n; i *= 2)
    for (int j = 1; j < n; j += i)
        for (int k = 1; k < j; k *= 2)
            // do something
```
2.
```
for (int i = 1; i < n; i *= 2)
    for (int j = n; j > 0; j /= i)
        // do something
```
3.
```
int count = 0;
for (int i = 1; i <= n; i++) {
    for (int j = i; j <= n; j += i) {
        count++;
    }
}
```
4.
```
for (int i = 1; i < n; i++)
    for (int j = 1; j < i; j *= j)
        // do something
```
5.
```
for (int i = 0; i < n; i++) {
    int j = i;
    while (j > 0) {
        j = j & (j - 1);
    }
}
```
6.
```
for (int i = 2; i < n; i *= i)
    for (int j = 1; j < sqrt(i); j++)
        // do something
```
7.
```
for (int i = 1; i < n; i *= 2)
    for (int j = 1; j < n; j *= i + 1)
        // do something
```
8.
```
for (int i = 1; i <= n; i++)
    for (int j = 1; j*j <= i; j++)
        for (int k = 1; k <= j; k++)
            // do something
```
9.
```
for (int i = 0; i < n; i++)
    for (int j = i; j > 0; j /= 2)
        for (int k = j; k > 0; k /= 2)
            // do something
```
10.
```
for (int i = 0; i < n; i += sqrt(i+1))
    // do something
```
11.
```
void fun(int n) {
    if (n <= 1) return;
    fun(n / 2);
    fun(n / 4);
```

```
    fun(n / 8);
}
12. void fun(int n) {
    if (n <= 1) return;
    fun(n - sqrt(n));
}
13. void fun(int n) {
    if (n <= 1) return;
    for (int i = 1; i < n; i++)
        fun(i);
}
14. void fun(int n) {
    if (n <= 1) return;
    fun(n / 2);
    fun(n / 2);
    for (int i = 0; i < n; i++) {}
}
15. void fun(int n) {
    if (n <= 1) return;
    fun(n / 2);
    fun(n / 3);
    fun(n / 6);
}
16. void fun(int n, int d) {
    if (n <= d) return;
    fun(n - 1, d);
    fun(n - 2, d);
}
17. void fun(int n) {
    if (n <= 1) return;
    for (int i = 0; i < log2(n); i++)
        fun(n / 2);
}
18. void fun(int n) {
    if (n <= 1) return;
    for (int i = 0; i < n; i += sqrt(i+1))
        fun(i);
}
19. void fun(int n) {
    if (n <= 1) return;
    fun(n / 2);
    for (int i = 0; i < n; i++)
        for (int j = 0; j < sqrt(i+1); j++)
            // do something
}
```

20.
```
void fun(int n) {
    if (n <= 1) return;
    for (int i = 1; i <= n; i *= 2)
        fun(n - i);
}
```
21.
```
vector<bool> prime(n, true);
for (int p = 2; p * p <= n; p++) {
    if (prime[p]) {
        for (int i = p * p; i < n; i += p)
            prime[i] = false;
    }
}
```
22.
```
for (int gap = n / 2; gap > 0; gap /= 2)
    for (int i = gap; i < n; i++)
        for (int j = i; j >= gap && arr[j] < arr[j - gap]; j -= gap)
            swap(arr[j], arr[j - gap]);
```
23.
```
for (int i = 0; i < n; i++) {
    int j = i;
    while (j < n) {
        j += __builtin_ctz(j + 1);
    }
}
```
24.
```
for (int i = 1; i <= n; i *= 2)
    for (int j = i; j <= n; j *= 3)
        for (int k = j; k <= n; k *= 5)
            // do something
```
25.
```
for (int i = 0; i < n; i++) {
    int x = i;
    while (x > 0) {
        x = x >> 1;
        for (int y = 0; y < x; y++)
            // do something
    }
}
```
26.
```
for (int i = 0; i < n; i++) {
    if (__builtin_popcount(i) % 2 == 0) {
        for (int j = 0; j < i; j++) {
            // do something
        }
    }
}
```
27.
```
for (int i = 0; i < n; i++) {
    for (int j = i; j <= n; j += __gcd(i + 1, j + 1))
        // do something
}
```

28.
```
int x = 1;
while (x < n) {
    x = pow(x, 1.5);
    // do something
}
```
29.
```
for (int i = 0; i < n; i++) {
    for (int j = 1; j <= i; j *= 2) {
        if ((i & j) != 0)
            // do something
    }
}
```
30.
```
for (int i = 0; i < n; i++) {
    int j = i;
    while (j) {
        j &= (j - 1);
        for (int k = 0; k < j; k++)
            // do something
    }
}
```