Time Complexity of Algorithm in DSA

1	`for(int i=1 ; i<=n ; i++ ){`
	`  // O(1)`
	`}`

2	`for(int i=1 ; 2i<=4n ; i++ ){`
	`  // O(1)`
	`}`

3	`for(int i=1 ; i<=n/3 ; i++ ){`
	`  // O(1)`
	`}`

4	`for(int i=1 ; 3i<=n ; i++ ){`
	`  // O(1)`
	`}`

5	`for(int i=1 ; 3^i<=n ; i++ ){`
	`  // O(1)`
	`}`

6	`for(int i=1 ; i<=n+100 ; i++ ){`
	`  // O(1)`
	`}`

7	`for(int i=1 ; i^3<=n ; i++ ){`
	`  // O(1)`
	`}`

8	`for(int i=n/2 ; i<=n ; i++ ){`
	`  // O(1)`
	`}`

9	`for(int i=1 ; i<=n ; i+=10 ){`
	`  // O(1)`
	`}`

10	`for(int i=n ; i>=1 ; i-- ){`
	`  // O(1)`
	`}`

11	`for(int i=1 ; i<=n ; i++ ){`
	`  for(int j=1 ; j<=i^2 ; j++ ){`
	`  // O(1)`
	`  }`

	}

12	for(int i=n ; i>=1 ; i-=5){
	  // O(1)
	}

13	for(int i=n ; i>=1 ; i/=2){
	  // O(1)
	}

14	for(int i=1 ; i<=1 ; i*=2){
	  // O(1)
	}

15	for(int i=1 ; i<=n ; i++){
	  for(int j=1 ; j<=n ; j++){
	  // O(1)
	  }
	}

16	for(int i=1 ; i<=n ; i++){
	  for(int j=1 ; j<=i ; j++){
	  // O(1)
	  }
	}

17	for(int i=1 ; i<=n ; i++){
	  for(int j=1 ; j<=100 ; j++){
	  // O(1)
	  }
	}

18	for(int i=1 ; i<=100 ; i++){
	  for(int j=1 ; j<=i ; j++){
	  // O(1)
	  }
	}

19	for(int k=1 ; k<=n ; k++){
	  for(int i=1 ; i<=n ; i++){
	  for(int j=1 ; j<=n ; j++){
	  // O(1)
	  }
	  }
	}

20	for(int k=1 ; k<=n ; k++){

	`for(int i=1 ; i<=i ; i++ ){`
	`  for(int j=1 ; j<=100 ; j++ ){`
	`  // O(1)`
	`  }`
	`  }`
	`}`

21	`for(int i=1 ; i<=n ; i++ ){`
	`  for(int j=1 ; j<=i^2 ; j++ ){`
	`  for(int k=1 ; k<=n/2 ; k++ ){`
	`  // O(1)`
	`  }`
	`  }`
	`}`

22	`for(int i=1 ; i<=n ; i++ ){`
	`  // O(1)`
	`}`
	`for(int i=1 ; i<=n^2 ; i++ ){`
	`  // O(1)`
	`}`

23	`for(int i=1 ; i<=n ; i++ ){`
	`  // O(1)`
	`}`

24	`for(int i=1 ; i<=n ; i++ ){`
	`  for(int j=1 ; j<=n ; j+=i ){`
	`  // O(1)`
	`  }`
	`}`

25	`for(int i=1 ; i<=n ; i*=2 ){`
	`  for(int j=1 ; j<=i ; j++ ){`
	`  // O(1)`
	`  }`
	`}`

26	`for(int i=n ; i>=1 ; i=  sqrt (i) ){`
	
	`  // O(1)`
	
	`}`

Answers in Details

1 –

```
     for(int i =1 i< n ; i++)
{
    //O(1)
}

since the above line of code have only one line of code to be
executed

so ;
        n * O(1) = n
        so the complexity of the above code is O(n)
```

2-

```
for(int i=1 ; 2i<=4n ; i++)
{
    //O(1)
}

since 2i is also dependent on the number of inputs "n" so 4 is
treated as
constant and we also knows that

    constant * O(n) = n

    so the whole complexity of this code is

    O(n)
```

3-

```
for(int i=1 ; i<=n/3 ; i++ ){
    // O(1)
}


the same as above the i is already dependent on the
number of inputs so

    the complexity is

                O(n)
```

4-

```
for(int i=1 ; 3i<=n ; i++ ){
    // O(1)
}


since i is dependent on number of inputs so

    the complexity is
    O(n)
```

5-

```
for(int i=1 ; 3^i<=n ; i++ ){
    // O(1)
}

since the repetitions of loop is dependent on the
number of inputs so the complexity is
        O(n)
```

6-

```
for(int i=1 ; i<=n+100 ; i++ ){
    // O(1)
}

since we know that O(n) + constant will
produces O(n)

    constant + O(n) = O(n)

    so the iterations dependent on the inputs so
    complexity will be
    O(n)
```

7-

```
for(int i=1 ; i^3<=n ; i++ ){
    // O(1)
}

since the repetitions of loop is dependent on the
number of inputs so the complexity is
        O(n)
```

8-

```
for(int i=n/2 ; i<=n ; i++ ){
    // O(1)
}


since the loop according to above code will
have to be executed by n/2 times
```

```
so 1/2 * n = n
so the complexity will be
O(n)
```

9-

```
for(int i=1 ; i<=n ; i+=10 ){
    // O(1)
}

since the loop has increment of 10 steps each times
so according to
constant + O(n) = n

the complexity will be O(n)
```

10-

```
for(int i=n ; i>=1 ; i-- ){
    // O(1)
}


simply changes the first complexity of increment into the
decrement
we can say that the complexity is also
    O(n)
```

11-

```
for(int i=1 ; i<=n ; i++ ){
    for(int j=1 ; j<=i^2 ; j++ ){
            // O(1)
  }
}
```

```
since the first loop will produces n
second loop generate n^2

so ;

    n^2 * n = n^3

    so complexity will be O(n^3)
```

12-

```
for(int i=n ; i>=1 ; i-=5 ){
    // O(1)
}

simply the complexity is O(n)
```

13-

```
for(int i=n ; i>=1 ; i/=2 ){
    // O(1)
}


since the program will produce the relation of iterations as
first iteration   = 1 = 1/(1/2)
second iteration  = 2 = 1/(1/4)
third iteration   = 3 = 1/(1/8)
.
.
.
.
n                 = n = 1/(1/2^n)

so this will give you O(logn)
```

14-

```
for(int i=1 ; i<=1 ; i*=2 ){
    // O(1)
}

the complexity is simply O(1)

because this loop will be run only one time
due to condition of less than or equal to 1
```

15-

```
for(int i=1 ; i<=n ; i++ ){
   for(int j=1 ; j<=n ; j++ ){
           // O(1)
   }
}


the first loop produces n
the second loop produces n

    n * n = n^2

    so the complexity is
        O(n^2)
```

16-

```
for(int i=1 ; i<=n ; i++ ){
   for(int j=1 ; j<=i ; j++ ){
           // O(1)
   }
```

```
}

first loop produces n
the second give n

    n * n = n^2

    so the complexity will be
      O(n^2)
```

17-

```
for(int i=1 ; i<=n ; i++ ){
   for(int j=1 ; j<=100 ; j++ ){
           // O(1)
   }
}


First loop will generate n
Second will give 1 because 100 is treated as constant

so constant * n = n

so the complexity is O(n)
```

18-

```
for(int i=1 ; i<=100 ; i++ ){
   for(int j=1 ; j<=i ; j++ ){
           // O(1)
   }
}

First loop will generate the constant
```

```
second will also give the constant because it is dependent on
the first loop

so
                constant * 1 = 1

                O(1)
```

19-

```
for(int k=1 ; k<=n ; k++ ){
   for(int i=1 ; i<=n ; i++ ){
      for(int j=1 ; j<=n ; j++ ){
             // O(1)
     }
  }
}

First loop will give    = n
Second loop will give   = n
Third loop will give    = n

so the complexity will be
                        O(n^3)
```

20-

```
for(int k=1 ; k<=n ; k++ ){
   for(int i=1 ; i<=i ; i++ ){
      for(int j=1 ; j<=100 ; j++ ){
             // O(1)
     }
  }
}
```

```
First loop will give    = n
Second loop will give   = 1
Third loop will give    = constant

        n * 1 * constant = n

        O(n)
```

21-

```
for(int i=1 ; i<=n ; i++ ){
   for(int j=1 ; j<=i^2 ; j++ ){
      for(int k=1 ; k<=n/2 ; k++ ){
             // O(1)
     }
  }
}

First loop generates = n
Second loop generates = n^2
Third loop generates = n/2 = 1/2 * n = n

               n * n * n^2 = n^4

               O(n^4)
```

22-

```
for(int i=1 ; i<=n ; i++ ){
    // O(1)
}
for(int i=1 ; i<=n^2 ; i++ ){
    // O(1)
}
```

```
In this case First loop will executed till n times
Second loop will be executed till n^2 times

since both loops will executed seperetely
so the equation will be

        n^2 + n = n^2

        because in case of sum of two values we will consider
the variable of higher order degree

        so
                O(n^2)
```

23-

```
for(int i=1 ; i<=n ; i++ ){
    // O(1)
}

As simple as earlier
        O(n)
```

24-

```
for(int i=1 ; i<=n ; i++ ){
   for(int j=1 ; j<=n ; j+=i ){
           // O(1)
   }
}

The First loop will generates = n
```

```
Second will iterate acording to = n + n because the increment
depends on i and it simply produces = 2n

        2n * n = 2 * n^2

        so
                o(n^2)
```

25-

```
for(int i=1 ; i<=n ; i*=2 ){
   for(int j=1 ; j<=i ; j++ ){
           // O(1)
   }
}


First loop will give n
Second will also give n

so
        o(n^2)
```

26-

```
for(int i=n ; i>=1 ; i=  sqrt (i) ){

           // O(1)

}

this loop will give sqrt(n)

                (n)^1/2
                it will also written as 1/2*log(n)
```

```
so O(logn)
```