**Note:** On the left side I have wrote the frequencies (only where it’s required), in the middle I have pasted the code and on the right side I have wrote the units each instruction will take.  
For sequential statements I have simply put -------- and wrote the unit it would take and for the loops I have used the bracket notation representing the units for each, for loop part respectively e.g. for(int I = 0; I < n; I ++) 🡨🡪 (1; n + 1; n). I have highlighted my answers with yellow. I hope this will help you to understand my way of solving.

**TASK – 01**

**1.**

- int a = 0, b = 0; -------- 1 unit

- for (i = 0; i < n; i++) ----- (1; n + 1; n)

- {

n a = a + i; -------- 1 unit

- }

- for (j = 0; j < n; j++) -------- (1; n + 1; n)

- {

n b = b + j; -------- 1 unit

- }

Solution: = 1 + 1 + n + 1 + n + n \* (1) + 1 + n + 1 + n + n \* (1)

T(n) = 6n + 5

**2.**

1 int a = 0; -------- 1 unit

- for (i = 0; i < n; i++) -------- (1; n + 1; n) unit

- {

n for (j = n; j > 0; j--)-------- (1, n + 1; n) unit

- {

n n a = a + i + j; -------- 1 unit

- }

- }

Solution: = 1 + 1 + n + 1 + n + n \* (1 + n + 1 + n + n)

T(n) = 3n2 + 4n + 3

**3.**

1 int sum = 0; -------- 1 unit

- for (int i = 1; i < n; i \*= 2) -------- (1; log2n + 1; log2n)

- {

log2n for (int j = 0; j < n; j++) -------- (1; n + 1; n)

- {

log2n n sum++; -------- 1 unit

- }

- }

Solution: = 1 + 1 + log2n + 1 + log2n + log2n \* (1 + n + 1 + n + n)

T(n) = 3n log2n + 4 log2n + 3

**4.**

1 int a = 0, i = n;

- while (i > 0) ------- (log2n + 2)

{

log2n + 1 a += i; ------- 1 unit

log2n + 1 i /= 2; ------- 1 unit

}

Solution: = 1 + log2n + 2 + (log2n + 1) \* (2)

T(n) = 3log2n + 5

**5.**

- for (int i = n; i > 0; i--) ------- (1; n + 1; n)

- {

n for (int j = 1; j < n; j \*= 2) ------- (1; log2n + 1; log2n)

- {

n log2n cout << i; ------- 1 unit

- }

- }

Solution: = 1 + n + 1 + n + n \* (1 + log2n + 1 + log2n + log2n)

T(n) = 3nlog2n + 4n + 2

**TASK – 02**

**1.**

1 int i, j, k = 0; ------- 1 unit

- for (i = n / 2; i <= n; i++) ------- (1; n / 2 + 1; n / 2)

- {

n / 2 for (j = 2; j <= n; j = j \* 3) ------- (1; log3n + 1; log3n)

- {

n / 2 log3n k = k + n / 2; ------- 1 unit

- }

- }

Solution: = (n / 2) \* (log3n)

Big – Oh = O(nlog3n)

**2.**

1 int count = 0; ------- 1 unit

- for (int i = n / 2; i <= n; i++) ------- (1; n/2 + 1; n/2)

n/2 for (int j = 1; j <= n; j = 2 \* j) ------- (1; log2n + 1; log2n)

n/2 log2n for (int k = 1; k <= n; k = k \* 2) -------- (1; log2n + 1; log2n)

n/2 log2n log2n count++; ------- 1 unit

Solution: = (n / 2 ) \* (log2n) \* (log2n)

Big – Oh = O(n(log2n)2)

**3.**

1 int count = 0; ------- 1 unit

- for (int i = n; i > 0; i /= 2) ------- (1; log2n + 1; log2n)

- {

log2n for (int j = 0; j < i; j++) ------- (1; i + 1; i)

- {

log2n count++; ------- 1 unit

- }

- }

Solution: = (log2n) \* ()

Big – Oh = O()\*(log2n))

**TASK – 03**

**1.**

- void function(int n)

- {

1 if (n == 1) ------- 1 unit

1 return; ------- 1 unit

- for (int i = 1; i <= n; i++) ------- (1; n + 1; n)

- {

n for (int j = 1; j <= n; j++) ------- (1; n + 1; n)

- {

n cout << "\*"; ------- 1 unit

n 1 break; ------- 1 unit

- }

- }

- }

Best – Case = Omega = Ω(2)

Worst – Case = Big Oh = O(n) //Because of break statement

**2.**

void function(int n)

{

if (n % 2 == 0) ------- 1 unit

{

for (int i = 1; i < n; i \*= 2) ------- (1; log2n + 1; log2n)

{

log2n cout << i << endl; ------- 1 unit

}

}

else

{

for (int i = 0; i < n; i++) ------- (1; n + 1; n)

{

n for (int j = 0; j < i; j++) ------- (1; i + 1; i)

{

n n cout << i + j << endl; ------- 1 unit

}

}

}

}

Best – Case = Omega = Ω(3) // T(n) = 3log2n + 3, put n = 1.

Worst – Case = Big Oh = O(n\* ))

**TASK – 04**

int main()

{

int num; ------- 1 unit

char ch; ------- 1 unit

cout << "Enter a number: "; ------- 1 unit

cin >> num; ------- 1 unit

cout << "If you wish to check whether the input number is "; ------- 1 unit

cout << "prime or not? enter 1"; ------- 1 unit

cout << "enter 2 to see fibonacci series till the input number."; ------- 1 unit

cin >> ch; ------- 1 unit

if (ch == '1') ------- 1 unit

{

if (isPrime(num) == true) ------- 3n1/2 + 1 unit

cout << num << " is a Prime number" << endl; ------- 1 unit

else

cout << num << " is not a Prime number" << endl; ------- 1 unit

}

if (ch == '2') ------- 1 unit

{

printFibonacci(num); ------- (6n – 2) unit

}

return 0; ------- 1 unit

}

void printFibonacci(int n)

{

int fib1 = 0; ------- 1 unit

int fib2 = 1; ------- 1 unit

for (int i = 2; i <= n; i = i + 1) ------- (1; n; n -1)

{

int temp = fib1 + fib2; ------- 1 unit

fib1 = fib2; ------- 1 unit

fib2 = temp; ------- 1 unit

cout << temp << endl; ------- 1 unit

}

}

bool isPrime(int n)

{

for (int i = 2; i <= sqrt(n); i++) ------- (1; sqrt(n); sqrt(n) - 1) unit

{

if (n % i == 0) ------- 1 unit

{

return false; ------- 1 unit

}

}

return true; ------- 1 unit

}

1. printFibonacci : = 2 + 1 + n + n – 1 + 4n – 4 = 6n – 2
2. isPrime : = 1 + n1/2 + n1/2 – 1 + n1/2 – 1 + 1 = 3n1/2
3. main : = 9 + 1 + 6n – 2 + 1 => T(n) = 6n + 9