CCI - Assignment # 2

SKIM THROUGH THESE 2 Pages, and everything should make sense.

SOME USEFUL Identities

(Make sure that every identity make sense to you)

1. Arithmetic Sequence Size

(There are how many terms in this sequence?)

1.1. $\Omega(N) = N/2 \leq \lvert \{ 1, 2, 3, 4, 5, 6, \ldots , N \} \rvert \leq N = O(N)$ # of terms are $\Theta(N)$	for(int i=1; i<=N; i++);
1.2. $\Omega(N) = N/4 <= \lvert \{ 0, 2, 4, 6, 8, 10, \ldots , N \} \rvert \leq N/2 = O(N)$ # of terms are $\Theta(N)$	for(int i=0; i<=N; i+=2);
1.3. $\Omega(N) = N/4 <= \lvert \{1, 3, 5, 7, \ldots , N\} \rvert \leq N/2$ i.e O(N) # of terms are $\Theta(N)$	for(int i=1; i<=N; i+=2);
1.4. $\Omega(N) = N/6 <= \lvert \{ 1, 4, 7, 10, \ldots , N \} \rvert \leq N/3$ i.e. O(N) # of terms are $\Theta(N)$	for(int i=1; i<=N; i+=3);
1.5. $\Omega(N) = \lvert \{1, 1+k, 1+2k, 1+3k, 1+4k, 1+5k, \ldots , N\} \rvert \leq N/k$ i.e. O(N) if k is a constant	for(int i=1; i<=N; i+=k);
1.6. $\Omega(N/\log N) = \lvert \{1, 1+\log N, 1+2 \log N, 1+3 \log N, 1+4\log N, 1+5 \log N, \ldots , N\} \rvert \leq N/\log N$ i.e. O(N/log N)	K = log N; for(int i=1; i<=N; i+=k);
1.7. $\Omega(\sqrt{N}) = \lvert \{1, 1+\sqrt{N}, 1+2\sqrt{N}, 1+3\sqrt{N}, 1+4\sqrt{N}, 1+5\sqrt{N}, \ldots , N \} \rvert \leq N/\sqrt{N} = O(\sqrt{N})$ i.e. $O(\sqrt{N})$	K = $\sqrt{N}$; for(int i=1; i<=N; i+=k);

(int i=1; i<=N; i+=10); N/10 times Similarly for(int i=1; i<=N; i+=20); N/20 times

for(int i=1; i<=N; i+=$\sqrt{N}$); $N/\sqrt{N} = \sqrt{N}$ ====> $N = \sqrt{N}.\sqrt{N}$

2. Arithmetic Series and relatives

Applications of $1+2+3+4+\ldots+N = \frac{N(N+1)}{2}$ If you don't remember this formula.

Proof

Upper Bound 1+2+3+4+5+6+ ....(N-3)+ (N-2)+ (N-1)+ N/2 <= N + N + N + ... + N replacing every value by the highest term
<= N x N = N^2 = O(N^2)

Lower Bound ~~1+2+3+4+5+6+...+ N/2+~~ (N/2+1)+(N/2+2) ....(N-3)+ (N-2)+ (N-1)+ N ignoring the first half
>= (N/2+1)+(N/2+2) + ... +(N-3)+ (N-2)+ (N-1)+ N and now replacing with the smallest value
>= N/2 + N/2 + N/2 + N/2 + ... + N/2 + N/2 + N/2 = N/2 x N/2 = ¼ N^2 = $\Omega(N^2)$
f(N) = $\Theta(N^2)$

2.1 $\Omega(T2) \leq 1+2+3+4+5+6+ \ldots T-3+ T-2+ T-1+ T \leq O(T^2)$ ====> $\Theta(T^2)$	
2.2 $\Omega(N^2) \leq 1+2+3+4+5+6+ + \ldots + N/2+N/2+1+ \ldots +N-3+ N-2+ N-1+ N \leq O(N^2)$ ===> $\Theta(N^2)$ $\Theta(N)$ $\Theta(N^2)$	for(int i=1; i<=N; i++) for(int j=1; j<=i; j++)
2.3 $\Omega(N^2) \leq 1+2+3+4+5+6+ \ldots (N/2-3)+ (N/2-2)+ (N/2-1)+ N/2 \leq O(N^2)$ ====> $\Theta(N^2)$ $\Theta(N)$ $\Theta(N^2)$	for(int i=1; i<=N; i+=2) for(int j=1; j<=i; j++)
2.4 $\Omega(N^2) \leq 1+2+3+4+5+6+ \ldots (N/3-3)+ (N/3-2)+ (N/3-1)+ N/3 \leq O(N^2)$ ====> $\Theta(N^2)$ $\Theta(N)$	for(int i=1; i<=N; i+=3)

$_{(1)}1+_{(1,2)}2+_{(1,2,3)}3+_{(1,2,3,4)}4+\ldots\ldots+_{(1,2,3,4\ldots,N/3)}N/3$ $\Theta(N^2)$	for(int j=1; j<=i; j++)
2.5 $\Omega(N) \le 1+2+3+4+5+6+ \ldots. +\sqrt{N}$ <= $O((\sqrt{N})^2)$ $\le O(N)$ ====> $\Theta(N)$ $\Theta(N^{1/2})$ $_{(1)}1+_{(1,2)}2+_{(1,2,3)}3+_{(1,2,3,4)}4+\ldots\ldots+_{(1,2,3,4\ldots,N^{1/2})}N^{1/2}$ $\Theta(N^2)$	for(int i=1; i<=$N^{1/2}$; i+=1) for(int j=1; j<=i; j++)

2.6 $\Omega((log\ N)^2) \le 1+2+3+4+5+6+ \ldots. +log\ N$ <= $O((log\ N)^2) \le \Theta(\log^2 N)$
$_{(1)}1+_{(1,2)}2+_{(1,2,4)}3+_{(1,2,3,4)}4+\ldots\ldots+_{(1,2,4,8\ldots,N)}\log N = \Theta(\log^2 N)$ ===>
$_{(1)}1+_{(1,2)}2+_{(1,2,3)}3+_{(1,2,3,4)}4+\ldots\ldots+_{(1,2,4,8\ldots,N)}\log N$
	for(int i=1; i<=N; i*=2) $\Theta(\log N)$
	for(int j=1; j<=i; j*=2)
	Example 2
	for(int i=1; i<=log N; i++)
	for(j=1; j<=i; j++) ; _

2.7 $\Omega(N^4) \le 1+2+3+4+5+6+ \ldots. + N^2$ $\le O(N^4)$ ====> $\Theta(N^4)$
$_{(1)}1+_{(1,2)}2+_{(1,2,3)}3+_{(1,2,3,4)}4+\ldots\ldots+_{(1,2,3,4\ldots,N^2)}N^2 = \Theta(N^4)$ ===>
	for(int i=1; i<=N*N; i++)
	$\Theta(N^2)$
	for(int j=1; j<=i; j++)

2.8 $\Omega(N^6) \le 1+2+3+4+5+6+ \ldots. + N^3 \le O(N^6)$ $\Theta(N^3)$ $_{(1)}1+_{(1,2)}2+_{(1,2,3)}3+_{(1,2,3,4)}4+\ldots\ldots+_{(1,2,3,4\ldots,N^3)}N^3$ $\Theta(N^6)$	for(int i=1; i<=N*N*N; i++) for(int j=1; j<=i; j++)
2.9 $\Omega(N^{2k}) \le 1+2+3+4+5+6+ \ldots. +N^k$ <= $O(N^k x N^k)$	
2.10 $\Omega(N^3) \le 1^2+2^2+3^2+4^2+5^2+6^2+ \ldots. + N^2 \le O(N^3)$ $_{(1)}1+_{(1,2,3,4)}2+_{(1,2,3,\ldots,9)}9+_{(1,2,3,4\ldots,16)}16+\ldots\ldots+_{(1,2,3,4\ldots,N^2)}N^2 = \Theta(N^3)$	for(int i=1; i<=N; i++) $\Theta(N)$ for(int j=1; j<=i*i; j++)
2.11 $\Omega(N^4) \le 1+2^3+3^3+4^3+5^3+6^3+ \ldots. + N^3 \le O(N^4)$ $\Theta(N)$ $_{(1)}1+_{(1,2,3,\ldots,8)}8+_{(1,2,3,\ldots,27)}27+_{(1,2,3,4\ldots,64)}64+\ldots\ldots+_{(1,2,3,4\ldots,N^3)}N^3$ $\Theta(N^4)$	for(int i=1; i<=N; i++) for(int j=1; j<=i*i*i; j++)
2.12 $\Omega(N^{k+1}) \le 1^k+2^k+3^k+4^k+5^k+6^k+ \ldots. +N^k$ <= $O(N^{k+1})$	

3. Some Examples

$\sqrt{N} * \sqrt{N} = N$ for(int i=1; i*i<=N; i++) Sum++; $O(\sqrt{N})$ for(int i=1; i*i<=N*N; i++) Sum++; O(N)

for(int i=1; i*i*i<=N*N; i++) Sum++; $O(N^{2/3})$ for(int i=1; i*i*i<=N; i++) Sum++; $O(N^{1/3})$

4. Geometric Sequence Size

4.1. $|\{N, N/2, N/4, N/8, N/2^4, N/2^5, N/2^6, \ldots 8, 4, 2, 1\}| = \log_2 N$

for(int i=1; i<=N; i*=2) or for(int i=N; i>=1; i/=2)

4.2. $|\{N, N/3, N/9, N/27, N/3^4, N/3^5, N/3^6, \ldots, 3^3, 9, 3, 1\}| = \log_3 N$

for(int i=1; i<=N; i*=3) or for(int i=N; i>=1; i/=3)

4.3. $|\{N, N/5, N/25, N/125, N/5^4, N/5^5, N/5^6, \ldots, 5^3, 5^2, 5, 1\}| = \log_5 N$

for(int i=1; i<=N; i*=5) or for(int i=N; i>=1; i/=5)

4.4. $|\{N, N/k, N/k^2, N/k^3, N/k^4, N/k^5, N/k^6, \ldots, k^3, k^2, k, 1\}| = \log_k N$

for(int i=1; i<=N; i*=2) or for(int i=N; i>=1; i/=k)

O(\logN)

Some formulas,

1. Log N^k = k log N	2. log N^2 = 2 log N = log N + log N	3. $\log^2 N$ = log N . log N

5. GEOMETRIC SERIES O(by the largest term)

Any Geometric Series with multiplicand factor of greater than 2(if increasing) or smaller then ½ (for decreasing geometric series) is bounded above and below by the largest term. If it is an increasing Geometric series in that case it will be the last term, if it is a decreasing series it will be the first term. For any constant - ratio(multiplication factor greater than 2 the above inequality is valid).

5.1. $\Omega(N) = N < 1+2+4+8+16+32+\ldots +N/4+N/2+N < 2N = O(N) \implies \Theta(N)$

```
for(int i=1; i<=N; i*=2)
        for(int j=1; j<=i; j++)
```

5.2. $\Omega(N) = N < 1+3+9+3^3+3^4+3^5+\ldots +N/3^2+N/3+N < 2N = O(N) \implies \Theta(N)$

```
for(int i=1; i<=N; i*=3)
        for(int j=1; j<=i; j++)
```

5.3. $\Omega(N^2) = N^2 < 1+3+9+3^3+3^4+3^5+\ldots +N^2/3^2+N^2/3 + N^2 < 2N^2 = O(N^2) \implies \Theta(N^2)$

```
for(int i=1; i<=N*N; i*=5)
        for(int j=1; j<=i; j++)
```

5.3. $N^3 < 1+5+5^2+5^3+5^4+5^5+\ldots +N^3/5^2+N^3/5+N^3 < 2N^3 = O(N^3)$

$1+5+5^2+5^3+5^4+5^5+\ldots +N^3/5^2+N^3/5+N^3 < 2N^3$

```
for(int i=1; i<=N*N*N; i*=5)
        for(int j=1; j<=i; j++)
```

5.4. $N^{3/2} < 1+5+5^2+5^3+5^4+5^5+\ldots +N^{3/2}/5^2+N^{3/2}/5+N^{3/2} < 2N^{3/2}$

```
for(int i=1; i<=N^(3/2); i*=2 )
        for(int j=1; j<=i; j++)
```

5.5. $N^{1/2} < 1+5+5^2+5^3+5^4+5^5+\ldots +N^{1/2}/5^2+N^{1/2}/5+N^{1/2} < 2N^{1/2}$

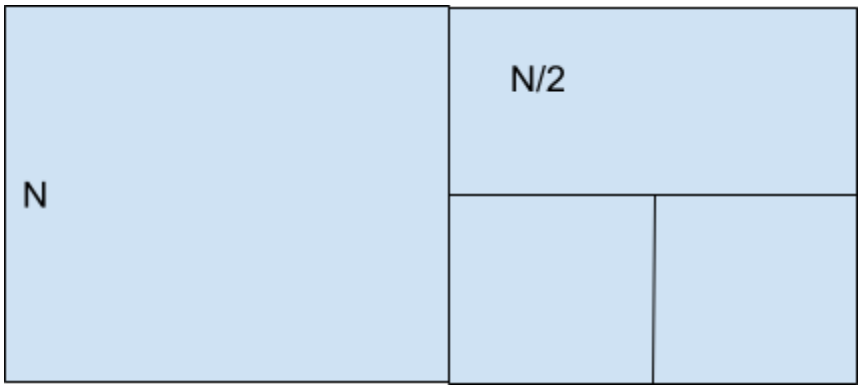
```
for(int i=1; i<=N^(1/2); i*=2 )
        for(int j=1; j<=i; j++)
```

5.6. $N < 1+5+5^2+5^3+5^4+5^5+\ldots +N/5^2+N/5+N < 2N$

$\log_a N = \frac{\log_b N}{\log_b a} \implies \log_2 N = \log_{10} N / \log_{10} 2 \implies \log_2 N = 1/\log_{10} 2 \cdot \log_{10} N$

Proof of Geometric Series?

$\Omega(N) = N < 1+2+4+8+16+32+\ldots +N/4+N/2+N < 2N = O(N) \implies \Theta(N)$



Part 1 - TIME COMPLEXITY

Note - This part shouldn't take more than 2 hours for this (actually it should take 41 mins).

If you are having any difficulty in finding out the complexity of the codes - read this document. It has many practice problems regarding time complexity with their solution mentioned in the comments. Also if you need further explanation(of the above sample practice problems), please watch my following VIDEO-LECTURE, which I delivered in my discrete mathematics course.

You have to give the time complexity of each line and do not just right answer justify the way it is done in part 1.

What is the algorithm's complexity of the following piece of code - Sample Solution is in RED.

```
int Sum=0;                  // O(1) Time
for(int i=0; i<N; i++)      //(1+1+1+...+1 - - - N Times =O(N)
  for(int j=0; j<N; j++)  Sum++;
  // (1+1+1+...+1) + (1+1 +... +1)+... + (1+1 +... +1) added N times
       // N          +      N       +... +       N          = O(N^2)
Overall Complexity : O(1) +O(N) + O(N^2) + O(N^2) = O(N^2)
```

2) What is the algorithm's complexity of the following piece of code

```
int Sum = 0;
for(int i = 1; i <= N; i *= 2)
   Sum++;
for(int j = 1; j <= N; j++){
   for(int k = 0; k < j; k++)
      Sum++;
   for(int m = 1; m <= j; m *= 2)
      Sum++;
}
```

3) What is the algorithm's complexity of the following piece of code

```
int Sum = 0;
for(int i = 1; i <= N; i *= 2)
   for(int j = 0; j < i; j++)
      for(int k = 1; k <= j+1; k *= 2)
         Sum++;

for(int x = 1; x <= N; x++)
   for(int y = x; y <= N; y++)
      for(int z = 1; z <= N; z *= 2)
         Sum++;
```

4) What is the algorithm's complexity of the following piece of code

```
int Sum = 0;
for(int i = 0; i < N; i++){
   for(int j = 0; j <= i; j++)
      Sum++;
}
for(int k = 0; k < N; k++){
   for(int m = 0; m < N-k; m++)
      Sum++;
}
for(int x = 0; x < N; x++){
   for(int y = 0; y < x; y++)
      for(int z = 0; z < y; z++)
         Sum++;
}
```

5)

```
int Sum=0;
for(int i=0; i<N; i++)
  for(int j=0; j<i; j++)
    for(int k=0; k<j; k++)
      Sum++;
```

6)

```
iint Sum = 0;
for(int i = 0; i < N; i += 2){
   for(int j = i; j < N; j += 3){
      for(int k = 0; k <= j; k += 2){
         for(int m = 0; m < k; m++)
            Sum++;
```

7

```
int Sum = 0;
for(int i = 1; i <= N; i *= 2){
   for(int j = 1; j <= i; j *= 2)   {
      for(int k = 0; k < j; k++)
         Sum++;
   }
}
```

8

```
int Sum=0;
for(int i=1; i<N; i*=2)
 Sum++;
for(int j=1; j<N; j*=2)
  Sum++;
```

9

```
for(int i = 1; i <= N*N; i += 2)
   for(int j = 1; j <= i; j *= 2)
      for(int k = 0; k < j; k++)
         for(int m = 1; m <= k+1; m *= 2)
            Sum++;
```

10

```
for(int i = 1; i <= N*N; i += 2)
   for(int j = 1; j <= i; j *= 2)
      for(int k = 0; k < j; k++)
         Sum++;
```

11

```
for(int i = 1; i <= N*N; i *= 2){
   for(int j = 1; j <= i; j *= 2){
      for(int k = 0; k < j; k++){
         for(int m = 1; m <= k+1; m *= 2)
            Sum++;
      }
   }
}
```

12

```
for(int i = 1; i <= N*N; i *= 2){
   for(int j = 1; j <= i; j *= 2)    {
      for(int k = 0; k < j; k++)
         Sum++;
   }
}
for(int x = 1; x <= N*N; x *= 2){
   for(int y = 0; y < x; y++)
      Sum++;
}
```

13

```
int Sum = 0;
for(int i = 1; i <= N; i *= 2)
   for(int j = 1; j <= i; j *= 2)
      for(int k = 0; k < j; k++)
         for(int m = 1; m <= k+1; m *= 2)
            Sum++;
```

14

```
int Sum = 0;
for(int i = 1; i <= N; i *= 2)
   for(int j = 1; j <= i; j *= 2)
      for(int k = 0; k < j; k++)
         Sum++;
```

15

```
int sum = 0;
for(int i = 1; i <= n; i *= 2)
   for(int j = 0; j < i; j++)
      for(int k = 1; k <= j+1; k *= 2)
         sum++;
```

16

BE CAREFUL GEOMETRIC SERIES

```
int sum,i,j;
sum = 0;
for (i=1; i<n; i=i*2)
   for (j=0; j <i ; ++j)
      sum++;
```

17

BE CAREFUL GEOMETRIC SERIES

```
int sum,i,j;
sum = 0;
for (i=1; i<n; i=i*5)
   for (j=0; j<i; j+=2)
      sum++;
```

18

```
int sum = 0;
for(int i = 1; i < n; i *= 4)
   for(int j = 0; j < i; j++)
      for(int k = 1; k <= j+1; k *= 2)
         sum++;
```

19 What will be the output (the value of Sum) of the program asymptotically in BIG-O notation, I am not asking here the complexity of loop rather the asymptotic bound on the value of Sum:

```
int Sum = 0;
for(int i=1; i<=n; i+=1)
  Sum+=i;
cout<<Sum<<endl;
```

20 What will be the output(the value of Sum) of the program asymptotically in BIG-O notation:

```
int Sum = 0;
for(int i=1; i<=n; i*=2)
  Sum+=i;
cout<<Sum<<endl;
```

21 What is the time complexity of the algorithm:

```
int Sum = 0;

for(int i = 1; i <= n; i++)
   for(int j = 1; j <= i; j++)
      Sum += j;
cout << Sum << endl;
```

22 What is the time complexity of the algorithm:

```
int Sum = 0;
for(int i = 1; i < n; i *= 2)
   for(int j = 1; j <= i; j++)
      for(int k = 1; k <= j; k *= 2)
         Sum++;
cout << Sum << endl;
```

23 What is the time complexity of the algorithm:

```
int f1(int n)
{
        int K=0;
          for(int j=0; j*j<=n*n; j++)  K++;
          return K;
}
int main()
{
        int Sum = 0,  n;
        cin>>n;
        for(int i=1; i<=f1(n); i+=1)
          for(int j=1; j<=i; j++)    Sum++;
        cout<<Sum<<endl;
}
```

24
What is the time complexity of the algorithm:

```
int f1(int n){
  int K=0;
  for(int j=1; j*j<=n; j*=2)       K++;
  return K;
}
int main()
{
int Sum = 0;
int n;
cin>>n;
for(int i=1; i<=f1(n); i+=1)
     for(int j=1; j<=i; j++)        Sum++;
cout<<Sum<<endl;
}
```

25 What is the time complexity of the algorithm:

```
int f1(int n){
   int K = 0;
   for(int i = 1; i * i <= n; i++){
      for(int j = 1; j <= i; j++)
         K++;
   }
   return K;
}

int main(){
   int Sum = 0;
   int n;
   cin >> n;
   int Terminator = f1(n);
   for(int i = 1; i <= Terminator; i++){
      for(int j = 1; j <= i; j *= 2){
         Sum++;
      }
   }
   cout << Sum << endl;
}
```

26 What is the time complexity of the algorithm:

```
int f1(int n) {
      int K=0;
      for(int j=0; j*j<=n; j++)
            K++;
      return K;
}
int main() {
      int Sum = 0;
      int n;
      cin>>n;
      int Terminator = f1(n);
      for(int i=1; i<=Terminator; i+=1) {
        for(int j=1; j<=i; j++)  {
            Sum++;
        }
      }
     cout<<Sum<<endl;
}
```

27

```
int f1(int n){
   int K = 0;
   for(int i = 1; i * i <= n; i++)    {
      for(int j = 1; j <= i; j++)
         K++;
   }
   return K;
}
int main(){
   int Sum = 0;
   int n;
   cin >> n;
   int Terminator = f1(n);
   for(int i = 1; i <= Terminator; i++){
      for(int j = 1; j <= i; j *= 2){
         Sum++;
      }
   }
   cout << Sum << endl;
}
```

28

```
for (i = 1; i < n; i = i * 4){
   cout << i;
   for (j = 0; j < i; j = j + 2){
      for (k = 1; k <= j+1; k = k * 2){
         cout << k;
         sum++;
      }
   }
   cout << sum;
}
```

29

```
for (i=1;i<=n*n;++i)
{
        cout << i;
            Sum=0;
            for (j=1; j<=i; ++j)
            {
                        Sum++;
                        cout << i;
            }
            cout << Sum;
}
```

30

```
for (i=1;i<=n*n*n;++i)
{
        cout << i;
            Sum=0;
            for (j=1; j<=i; ++j)
            {
                        Sum++;
                        cout << i;
            }
            cout << Sum;
}
```

31

```
for (int i = 1; i <= n*n*n; i *= 2){
   cout << i;
   Sum = 0;
   for (int j = 1; j <= i; j++){
      for (int k = 1; k <= j; k *= 2){
         Sum++;
         cout << k;
      }
   }
   cout << Sum;
}
```

32

```
for (i=1;i<=n*n*n; i*=2){
        cout << i;
            Sum=0;
            for (j=1;j<=n; j++){
                        Sum++;
                        cout << i;
            }
            for (k=1;k<=n; k++){
                        Sum++;
                        cout << i;
            }
            cout << Sum;
}
```

33

```
for (int i = 1; i <= n*n*n; i += 2){
   cout << i;
   Sum = 0;
   for (int j = 1; j*j <= i; j++){
      for (int k = 1; k <= j; k++){
         Sum++;
      }
   }
   for (int x = 1; x <= n; x++){
      for (int y = 1; y*y <= x; y++){
         Sum++;
      }
   }
   cout << Sum;
}
```

34

```
for (int i = 1; i <= n*n*n; i++){
   Sum = 0;
   for (int j = 1; j*j <= i; j++){
      for (int k = 1; k <= n; k++){
         Sum++;
      }
   }
   for (int x = 1; x <= n; x++){
      for (int y = 1; y*y <= x; y++){
         Sum++;
      }
   }

   cout << Sum;
}
```

35-36

```
for (int i=1; i <= n ; i = i * 2)
{
            for ( j = 1 ; j <= i ; j = j * 2)
                        cout<<"*";
}
```

```
for (int i=1; i <= n ; i = i * 2)
            for ( j = 1 ; j <= i ; j = j * 2)
                        cout<<"*";

for (int i=1; i <= n ; i = i * 2)
            for ( j = 1 ; j <= i ; j = j * 2)
                        cout<<"*";
```

37

```
for (int i = 0; i < n; i = i + 3){
   cout << i;
   for (int j = 1; j <= i + 1; j++){
      for (int k = 1; k <= j; k = k * 3){
         sum++;
         cout << k;
      }
   }
   for (int x = 1; x <= i + 1; x++){
      sum++;
   }
   cout << sum;
}
```

38

```
for (int i=1; i <= n ; i = i * 2)
        for ( j = 1 ; j <= i ; j = j * 2)
            cout<<"*";

for(int i=0; i<=N; i++)
 Sum++;
```

39

```
for (int i = 0; i < n; i = i + 3)
  cout << i;
  for (int j = 1; j <= i + 1; j++)
     for (int k = 1; k <= j; k = k * 3)
        sum++;

 for (int x = 1; x <= n; x++)
    for (int y = 1; y <= x; y = y * 3)
       sum++;
 cout << sum;
```

40 Complexity of primeNumber function.

```
int sqrt(int N)
{
  int d;
  for(d=0; d*d<=N; d++) {   }
  return d-1;
}
bool primeNumber(int n)
{
        bool isPrime = true;
        int lmt = (sqrt(n));
        for (int d=2; d <=lmt ;++d)
        {
                if (n%d==0)
                        return false;
        }
        return true;
}
```

41 Complexity of primeNumber function.

```
int sqrt(int N)
{
  int d;
  for(d=0; d*d<=N; d++){ }
  return d-1;
}
bool primeNumber(int n)
{
        bool isPrime = true;
        for (int d=2; d <= sqrt(n) ;++d)
        {
                if (n%d==0)
                        return false;
        }
        return true;
}
```

PART 2 (Recursion)

Should Take Maximum 2 hours

1. **Write the recursive and iterative code for Fibonacci Number computation.**
   - I. Analyze why Iteration is working so fast as compared to recursive implementation of Fibonacci Numbers. Its written in the notes.
   - II. Run the recursive code on Fib(10), Fib(20), Fib(30), Fib(40), Fib(50), analyse what is happening?
   - III. Why it is shocking?

2. **Write the recursive and iterative code for Computing the TriSum sequence:** 1, 2, 3, 6, 11, 20, 37, ……..
   - I. Write the recursive mathematical formulation.
   - II. Write the recursive code for the Sequence generator
   - III. Run on TriSum(10), TriSum(20), TriSum(30), TriSum(40), TriSum(50) - got a shock?

3. **Write a recursive function for computing the following Series for each part. Drive their time complexity. (6+4) points**

I.	1+2+3+4+...+N	IV.	$1+3+9+27+81+...+3^N$
II.	1+3+5+7+....+N (where N must be odd)	V.	1+3+9+27+81+... + N/9 + N/3+N
III.	$1+2+4+8+16+...+2^N$ (N is an integer)	VI.	1+2+4+8+16+...+ N/2+N

4. **Write a recursive function using the recursive definition of Permutation, Combination (Formula). Using the following formulas (5+5) points**
   - I. $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$, **Also denoted as C(N, R) = C(N-1, R-1) + C(N-1, R) is combination. Define base case by yourself?**
   - II. **P(N, R) = N * P(N-1, R-1) and base case is P(N, R) = 1 if R=0.**

5. **Write the Recursive Binary Search and test it on a huge Data.**

6. Write the code for **Merge - Recursively** (in class we did iterative Merge), then write **MergeSort**.
    I. Prove its time complexity is O(N log N) in the worse case. Read from chatGPT - specially recursive analysis, by asking draw me the recursive tree and show how many levels it will take if we have like N = 2^k number of input array size, see why the number of levels will be O(log N) and why the overall complexity on each level will be O(N).

7. Write Segregate Recursively (in class we did iterative Merge), then write **QuickSort**.
    I. Prove its time complexity is O(N^2) in the worse case.
    II. Prove its time complexity in the best case if O(N log N). Read from chatGPT - specially recursive analysis, by asking draw me the recursive tree and show how many levels it will take if we always the pivot in the middle, see why the number of levels will be O(log N) and why the overall complexity on each level will be O(N).

8. Write a Recursive function to compute POWER(X, Y, M) compute X^Y % M (X^Y modulo M).
    I. **The algorithm must take O(Y) times additions/subtractions.**

[THE HACKER's Challenge]
    II. **The algorithm must take O(log Y) times additions/subtractions.**

9. Write a Program which wants to do multiplication of AxB imagine all A and B are n bit strings and there is module available **ADD(X,Y)** and you want to write this **MULT(X,Y)** using **ADD(X,Y)** How you will going to write this module. Write the module such that It takes a minimum number of steps, obviously Calling ADD Y times is very bad idea and unacceptable.
    I. The algorithm must take O(Y) times additions/subtractions.

[THE HACKER's Challenge]
    II. **The algorithm must take O(log Y) times additions/subtractions.**

10. Write a Program which you compute A/B and A%B and the only operations allowed are subtraction and addition.
    I. The algorithm must take O(B) times additions/subtractions.

[THE HACKER's Challenge]
    II. **The algorithm must take O(log B) times additions/subtractions.**

THE HACKER'S Challenge — These are the challenges that will require more time and thought. They are the **food-for-thought questions** that you keep thinking about until the next class. Don't just sit in front of your computer to solve them—think about them in your head while you're walking around, having a coffee, or simply going about your day.