Sallam

Factorial

Find Factorial of numbers from 1 to 20 in Time $O(n)$ and memory $O(n)$.

Code

```
1-  long Fact[21];
2-  void getFact() {
3-  Fact[0]=1;
4-  for(long i=1;i<=20;i++) { Fact[i]=i*Fact[i-1]; }
5-  }
```

Prime Factorization

Find prime factorization of number in Time $O(\sqrt{n})$ and memory $O(1)$.

Code

```
1-  void primeFactors(int n) {
2-      for(int i=2;i*i<=n;i++){
3-          while (n%i==0){
4-              cout<<i<<' ';
5-              n/=i;
6-          }
7-      }
8-      if(n>1){ cout<<n<<' '; }
9-  }
```

Another Code to Prime Factorization

This Approach is best for all composite numbers and achieves in Time $O(log(n))$ but is O(n) otherwise and memory $O(1)$.

Code

```
1-  void primeFactors(int n)
2-  {
3-      int c = 2;
4-      while (n > 1) {
5-          if (n % c == 0) {
6-              printf("%d ", c);
7-              n /= c;
8-          }
```

Sallam

```
9-          else
10-             c++;
11-     }
12- }
```

Divisor Algorithm

Given number n Find all divisors of n

Time Complexity: $O(log(n))$, Auxiliary Space: $O(1)$

Code

```
1-  void allDivisors(int n)
2-  {
3-      for(int i=1;i*i<=n ;i++)
4-      {
5-          if(n%i==0)
6-          {
7-              cout<<i<<' ';
8-              if(i!=n/i)cout<<n/i<<' ';
9-          }
10-     }
11- }
```

Prime Algorithms

Given number n Check if n is prime or not in

Time Complexity: $\sqrt{n}$,**Auxiliary Space**: $O(1)$

Code

```
1-  bool isPrime(int n)
2-  {
3-      if (n <= 1)
4-          return false;
5-
6-      for (int i = 2; i*i<= n; i++)
7-          if (n % i == 0)
```

Sallam

```
8-              return false;
9-      return true;
10- }
```

Sieve Of Eratosthenes Prime Algorithm

Given number n Check if n is prime or not in

Time Complexity: $O(\mathbf{N} * \log_2(\log_2(n)))$, **Auxiliary Space**: $O(n)$

Code

```
1-  void SieveOfEratosthenes(int n)
2-  {
3-      bool prime[n + 1];
4-      memset(prime, true, sizeof(prime));
5-      for (int p = 2; p * p <= n; p++) {
6-          if (prime[p] == true) {
7-              for (int i = p * p; i <= n; i += p)
8-                  prime[i] = false;
9-          }
10-     }
11- }
```

SPF

The Smallest Prime Factor (SPF) array Used to efficiently find the prime factors of a range of numbers.

```
1-  int spf[N];
2-
3-  void SPF() {
4-      iota(spf, spf + N, 0);
5-      spf[1] = 0;
6-      for (int i = 2; i < N; ++i) {
7-          if (spf[i] != i)continue;
8-          for (int j = 2 * i; j < N; j += i) {
9-              if (spf[j] == j)
```

Sallam

```
                spf[j] = i;
            }
        }
}
```

This is how to use.

```
vector<int> primeFactors(int n) {
    vector<int> factors;
    while (n > 1) {
        factors.push_back(spf[n]);
        n /= spf[n];
    }
    return factors;
}
```