MATHS

Q1) count digits

```
#include<bits/stdc++.h>
using namespace std;
void countdigits(int n){
   int count=0;
   while(n!=0){
      count++;
      n=n/10;
   }
   cout<<count;
}
int main(){
   int n;
   cin>>n;
   countdigits(n);
}
```

Q2)palindrome numbers

```
#include<bits/stdc++.h>
using namespace std;
void palindromenum(int n){
  int res=n;
  int tes=0;
  while(res!=0){
    int i=res%10;
    tes=tes*10+i;
    res=res/10;
  }
  if(tes==n){
     cout<<"is a palindrome";
  }else{
     cout<<"is not a palindrome";
  }
}
int main(){
  int n;
  cin>>n;
  palindromenum(n);
}
```

Q3)trailing zeros

Naïve approach:

```
#include<bits/stdc++.h>
using namespace std;
void trailingzeros(int n){
   int fact=1;
   for(int i=2;i<=n;i++){
      fact=fact*i;
   }
   int res=0;
   while(fact%10==0){
      res++;
      fact=fact/10;
   }
   cout<<res;
}
int main(){
   int n;
   cin>>n;
   trailingzeros(n);
}
```

Pro approach

```
#include<bits/stdc++.h>
using namespace std;
```

```
void trailingzeros(int n){
  int res=0;
  for(int i=5;i<=n;i=i*5){
     res=res+n/i;
  }
  cout<<res;
}
int main(){
   int n;
   cin>>n;
   trailingzeros(n);
}
```

Q4)gcd of two numbers

Naïve:

```
#include<bits/stdc++.h>
using namespace std;
void gcd(int m,int n){
  int res=min(m,n);
  while(res>0){
     if(m%res==0 && n%res==0){
        cout<<res;
        exit(0);
     }else{
        res--;
     }
  }
}
int main(){
  int m,n;
  cin>>m>>n;
  gcd(m,n);
}
```

Pro:(Euclidean algo)

```
#include<bits/stdc++.h>
using namespace std;
void euclidean(int a,int b){
  while(a!=b){
     if(a>b){
        a=a-b;
     }else{
        b=b-a;
     }
  }
  cout<<a;
}
int main(){
  int m,n;
  cin>>m>>n;
  euclidean(m,n);
}
```

Short cut for gcd:

```
#include<bits/stdc++.h>
using namespace std;
void shortcut(int a,int b){
  int x=__gcd(a,b);
  cout<<x;
}
int main(){
  int m,n;
  cin>>m>>n;
  shortcut(m,n);
}
```

Time Complexity: O(k*logn)

Auxiliary Space: O(k)

Q5)lcm of two numbers

naive

```
#include<bits/stdc++.h>
using namespace std;
void lcm(int m,int n){
  int res=max(m,n);
  while(res>0){
     if(res%m==0 && res%n==0){
        cout<<res;
        exit(0);
     }else{
        res++;
     }
  }
}
int main(){
  int m,n;
  cin>>m>>n;
  lcm(m,n);
}
```

Pro:

```
#include<bits/stdc++.h>
using namespace std;
void shortcut(int a,int b){
  int x=__gcd(a,b);
  int lcm=(a*b/x);
  cout<<lcm;
}
int main(){
  int m,n;
  cin>>m>>n;
  shortcut(m,n);
}
```

Q6)Check for prime

```
#include<bits/stdc++.h>
using namespace std;
void checkforprime(int n){
     if(n==1){
      cout<<"no";
      exit(0);
   }
   if(n==2 || n==3){
                        cout<<"yes";
   }
   if(n%2==0 && n%3==0){
      cout<<"no";
      exit(0);
   }
   for(int i=5;i*i<=n;i=i+6){//skipping 5 terms ahead
      if(n%i==0 || n%(i+2)==0){
         cout<<"no"<<endl;
         exit(0);
      }
      cout<<"yes"<<endl;
   }


}
int main(){
  int m,n;
  cin>>m;
  checkforprime(m);
}
```

Q7)prime factors

Naïve:

```
#include<bits/stdc++.h>
```

```
using namespace std;
int prime(int n){
      if(n==1){
        return false;
    }
    if(n==2 || n==3){
                          return true;
    }
    if(n%2==0 && n%3==0){
        return false;
    }
    for(int i=5;i*i<=n;i=i+6){//skipping 5 terms ahead
        if(n%i==0 || n%(i+2)==0){
            return false;
        }
        return true;
    }
    return 0;
}

void primefactors(int m){
    int x;
    for(int i=2;i<m;i++){
        if(prime(i)){
         x=i;
        }
        while(m%x==0){
            cout<<i;
            x=x*i;
        }
    }
}
int main(){
   int m,n;
   cin>>m;
   primefactors(m);
}
```

Pro method:

```
#include<bits/stdc++.h>
using namespace std;
 void primefactors(int n)
{
            if(n <= 1){
                        cout<<"not a prime";
                        exit(0);
            }

            for(int i=2; i*i<=n; i++){
                        while(n % i == 0){
                                    cout<<i<<" ";
                                    n = n / i;
                        }
            }
            if(n > 1)
                        cout<<n<<" ";
            cout<<endl;
}
int main(){
   int m,n;
   cin>>m;
   primefactors(m);
}
```

More efficient:

You can reduce the iterations from the check for prime method …

Q8) print divisors(in sorted order)

Naïve:

By just traversing the whole ..

Pro:

```
#include<bits/stdc++.h>
using namespace std;
 void divisors(int n){
        int i = 1;
        for(i=1; i*i < n; i++){
                if(n % i == 0){
                        cout<<i<<" ";
                }
        }
        for(; i >= 1; i--)     {
                if(n % i == 0){
                        cout<<(n / i)<<" ";
                }
        }
}
int main(){
  int m,n;
  cin>>m;
  divisors(m);
}
```

Q9)sieve of erosthenes

Prime numbers upto the range of the number

Naïve :

We traverse the whole n numbers and check whether the number is prime or not

Pro:

```
#include<bits/stdc++.h>
using namespace std;
 void sieve(int n){
        vector<bool>isprime(n+1,true);
        for(int i=2;i*i<=n;i++){
          if(isprime[i]){
            for(int j=2*i;j<=n;j=j+i){
              isprime[j]=false;
            }
          }
        }
        for(int i=2;i<=n;i++){
          if(isprime[i]){
            cout<<i;
          }
        }
}
int main(){
  int m,n;
  cin>>m;
  sieve(m);
}
```