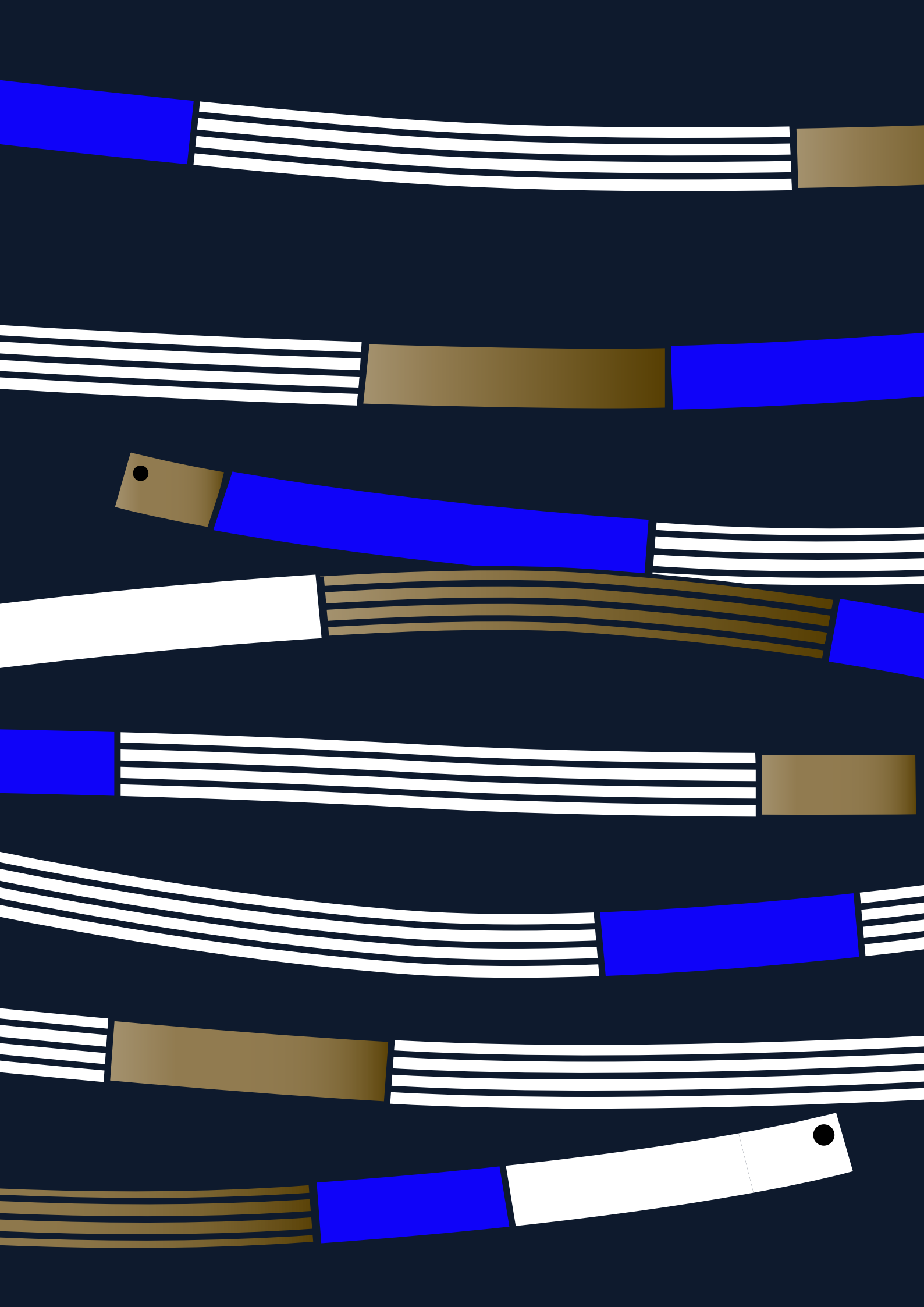

Given a number print all its factors

minimum factor of a number - 1

maximum factor of number N - N

```
int count=0;
for(int i=1 ; i<=N; i++){
    if(N%i ==0)
        count ++;
}
```

CPU clock rate - Giga Hertz (GHz) - power of a CPU with which it processes a task in one second.

Ex: 3.2 GHz Giga - 10^9

$3.2 * 10^9$ 10^9 - billion operations per second

When we say 3.2 GHz in a computer it simply means it can perform 3.2 GHz iterations per second. So when there are operations done in a program like count, if conditions - they will be calculated separately.

In an approximation,

1 sec - $3.2 * 10^9$ operations

⇓

1 sec - 10^8 iterations

converted by dividing by some approx. value

```
int count=0;
for(int i=1 ; i<=N; i++){
    if(N%i ==0)
        count ++;
}
```

if N is 10^9

By fact, we know

10^8 iterations - 1 sec

1 iteration - $\frac{1}{10^8}$ sec

∴ 10^9 iterations - $\frac{10^9}{10^8}$ sec

10^9 iterations - 10 sec

for $N = 10^{18}$,

10^{18} iterations - $\frac{1}{10^8} \times 10^{18}$ - 10^{10} seconds - 317 years

So, we can say we have to optimize our solution for any given number, we can only find $N/2$ factors.

Assume, $N = 110$ so we find $110/2 = 55$ factors because after 55, there would be only one factor and that would be the number N itself.

So, there would never be an individual number being a factor of N.

Ex:- 20 - one of the factor is 20

but to find it we would do $1*20, 5*4, 10*2$ etc, so, we can say → $i * j = N$

A factor always exist with a pair.

$i * j = N \Rightarrow j = N/i$

if i and j are 2 factors, i and N/i are factors as well

N = 24

i		$j = N/i$
1	<	24
2	<	12
3	<	8
4	<	6
6	>	4
8	>	3
12	>	2
24	>	1

this is nothing but same as above part written in reverse order.

N = 36

i		$j = N/i$	
1	<	36	
2	<	18	
3	<	12	
4	<	9	
6	==	6	→ we also might have same pair of factors
9		4	
12		3	
18		2	
36		1	

so we can count only the first part. This time since there is a pair we need to count twice and ignore the second half.

Other thing to note is i is less than j and $j = N/i$

i.e., $i < N/i$

Or $i*i < N$ would be our condition.

$i*i < N$

$i^2 < N$

$i < \sqrt{N}$

```
int count=0;
for(int i=1 ; i*i < N ; i++){
    if(N%i ==0)
        count +=2;
}
```

When pair of factors are same, we need to have a condition and count only once.

```
int count=0;
for(int i=1 ; i*i <=N ; i++){
    if (N%i ==0) {
        if (i == N/i)
            count +=1;
        count +=2;
    }
}
```

Time complexity - $O(\sqrt{n})$

if $N = 10^{18}$ - $\sqrt{10^{18}} = 10^9$

1 iteration - $\frac{1}{10^8}$

10^9 iterations - $\frac{1}{10^8} \times 10^9$

10^{18} iterations - 10 seconds

we reduced 317 years to 10 seconds after optimiza-tion.

2. Given a number N, check if the number is prime

Every prime number will only have 2 factors - 1 and the number itself.

Using the previous solution if the count is two we say it is prime otherwise not.

```
int count =0;
for(int i= 1;  i*i<=N; i++){
      if (N%i ==0){
            if (i == N/i){ count += 1;}
            count +=2;
      }
}
if(count ==2) { print ("prime");}
else { print("not prime");}
```