In [8]:

```
def to_binary(number):
    if number < 0:
        raise ValueError("Only positive integers are allowed")

    return bin(number)[2:]

# Test the function
user_input=int(input("enter the number"))
print(to_binary(user_input))  # Output: 1010
```

```
11101111
```

In [15]:

```
def find_factors(number):
    if number <= 0:
        raise ValueError("Only positive integers are allowed")
    factors = []
    for i in range(1, number + 1):
        if number % i == 0:
            factors.append(i)
    return factors

# Test the function
user_input=int(input("enter any positive integers"))
print(f"The factors of {user_input} are: {find_factors(user_input)}")  # Output: [1, 2, 3, 4, 6, 12]
```

```
The factors of 234 are: [1, 2, 3, 6, 9, 13, 18, 26, 39, 78, 117, 234]
```

In [24]:

```
def is_prime(number):
    if number <= 1:
        return False
    for i in range(2, int(number ** 0.5) + 1):
        if number % i == 0:
            return False
    return True

# Test the function
user_input=int(input("Enter any number greater than 1 = "))
print(is_prime(user_input))  # Output: True or false
```

```
False
```

In [25]:

```
def encrypt_message(message):
    encrypted = message.replace(" ", "")[::-1]
    return encrypted

# Test the function
user_text=str(input("Enter a message"))
print(encrypt_message(user_text))  # Output: opposite of the user_text
```

```
nrakannasrp
```

In [46]:

```
import random
import string

def random_encrypt(message):
    filler_length = random.randint(2, 5)
    encrypted_message = []

    for char in message:
        encrypted_message.append(char)
        for i in range(filler_length):
            encrypted_message.append(random.choice(string.ascii_letters))

    encrypted_string = ''.join(encrypted_message)
    return encrypted_string, filler_length


# Test the function

encrypted_code = input("Enter your secret message: ")
encrypted_message, filler_length = random_encrypt(encrypted_code)
print(f"The encrypted message is {encrypted_message} and the interval used is {filler_length}")
```

```
The encrypted message is bMLplQWTaNnGaNsIaqDRakJl and the interval used is 3
```

In [ ]:

```
def decrypt_message(encrypted_message, interval):
    decrypted_message = encrypted_message[::interval]
    return decrypted_message

# Test the function
encrypted_msg = "sxyexynxydxycxyhxyexyexysxye"
interval = 2
print(decrypt_message(encrypted_msg, interval))  # Output: "send cheese"
```