```
#1:
def int_to_bin(n):
    if n <= 0:
        return("Error: Input must be a positive integer.")
    return bin(n)[2:] #function that converts an integer to its binary
representation as a string
number = int(input("Enter the number for conversion into binary: "))
print(int_to_bin(number))
```

Enter the number for conversion into binary: 89

1011001

```
#2:
def int_factors(n):
    if n <= 0:
        print("Error: Input must be a positive integer.")
        return []  # Return an empty list

    factors = []
    for i in range(1, n + 1):
        if n % i == 0:
            factors.append(i)

    return factors
#testing:
number = int(input("Enter the number for factorisation: "))
print(int_factors(number))
```

Enter the number for factorisation: 12

[1, 2, 3, 4, 6, 12]

```
#3:
from math import sqrt
def is_prime(n):
    if n <= 1:
        return "Please enter value greater than 1."

    for i in range(2, int(sqrt(n)) + 1):
        if n % i == 0:
            return False  # n is divisible by i, so it's not prime
```

```
    return True  # n is prime if no divisors were found

#testing:
prime = int(input("Enter a number for checking if its a prime number: "))
print(is_prime(prime))
```

Enter a number for checking if its a prime number: 87

False

```
#4:
def encrypt_message(message):
    # The replace(" ", "") method is used to remove all spaces from the string.
This creates a new string no_spaces that contains the original message without
any spaces.
    no_spaces = message.replace(" ", "")
    # The slicing operation [::-1] is used to reverse the string. This creates a
new string encrypted that is the reverse of no_spaces.
    encrypted = no_spaces[::-1]
    return encrypted

message = input("Type string for encryption: ")
print(encrypt_message(message))
```

Type string for encryption: weeklytasks

sksatylkeew

```
#5

#5:
import random
import string

def encrypt_with_random_letters(message):
    # Generate a random interval between 2 and 20
    interval = random.randint(2, 20)

    # Create a list to hold the encrypted message
```

```
    encrypted_message = []
    key = []  # Store all the original characters that are replaced

    # Iterate through the message and fill in the gaps
    for i in range(len(message)):
        if (i % interval) == 0:  # If the index is a multiple of the interval
            encrypted_message.append(message[i])  # Add the character from the
message
        else:
            # Add a random letter from the alphabet
            random_letter = random.choice(string.ascii_lowercase)
            encrypted_message.append(random_letter)
            key.append((i, random_letter))  # Store the position and random
letter in the key

    # Join the list into a single string
    encrypted_message_str = ''.join(encrypted_message)

    return encrypted_message_str, interval, key

# Example usage
message = "encryptiondecryption"
encrypted_message, interval, key = encrypt_with_random_letters(message)
print("Encrypted Message:", encrypted_message)
print("Interval:", interval)
print("Key:", key)


#  testing
message = input("Enter the message for encryption: ")
encrypted_message, interval, key = encrypt_with_random_letters(message)

print(f"Original Message: {message}")
print(f"Encrypted Message: {encrypted_message}")
print(f"Interval: {interval}")
print(f"Key: {key}")
```

Encrypted Message: ekadchmgvyrkmlykkbtr

Interval: 14

Key: [(1, 'k'), (2, 'a'), (3, 'd'), (4, 'c'), (5, 'h'), (6, 'm'), (7, 'g'), (8, 'v'), (9, 'y'), (10, 'r'), (11, 'k'), (12, 'm'), (13, 'l'), (15, 'k'), (16, 'k'), (17, 'b'), (18, 't'), (19, 'r')]

Enter the message for encryption: shortestlongest

Original Message: shortestlongest

Encrypted Message: sidfmlrphogcwus

Interval: 9

Key: [(1, 'i'), (2, 'd'), (3, 'f'), (4, 'm'), (5, 'l'), (6, 'r'), (7, 'p'), (8, 'h'), (10, 'g'), (11, 'c'), (12, 'w'), (13, 'u'), (14, 's')]

```
#6:
def decrypt_with_random_letters(encrypted_message, interval, key):
    decrypted_message = []
    key_index = 0

    for i in range(len(encrypted_message)):
        if i % interval == 0:
            decrypted_message.append(encrypted_message[i])  # Use the retained
character
        else:
            decrypted_message.append(key[key_index % len(key)])  # Cycle through
the key
            key_index += 1

    return ''.join(decrypted_message)

encrypted_message = input("Enter the encrypted message: ")
interval = int(input("Enter the encryption interval: "))
key = input("Enter the key (comma-separated): ").replace("(", "").replace(")",
"").split(',')

original_message = decrypt_with_random_letters(encrypted_message, interval, key)

print(f"Encrypted Message: {encrypted_message}")
print(f"Decrypted (Original) Message: {original_message}")
```

Enter the encrypted message: randomletters

Enter the encryption interval: 7

Enter the key (comma-separated): 1,2

Encrypted Message: randomletters

Decrypted (Original) Message: r121212e12121