Q(11) Accept a filename from user and print its extension.

```
# Accept a filename from the user
filename = input("Enter a filename: ")

# Check if the input contains a dot
if "." in filename:
    # Split the filename into name and extension
using the split() method
    name, extension = filename.split(".")

    # Print the extension
    print("The extension of the file is: " +
extension)
else:
    print("Invalid filename format. Please include
the file extension (e.g., filename.txt)")
```

Q(12) Create a list of colors from comma-seperated color names entered by user. Display first and last colors.

```
# Accept comma-separated color names from the user
color_names = input("Enter comma-separated color names: ")

# Split the input string into a list of color names
colors = color_names.split(',')
```

```
# Remove leading and trailing whitespaces from each color name
colors = [color.strip() for color in colors]

# Check if the user entered at least two colors
if len(colors) >= 2:
    # Display the first and last colors
    print("First color: ", colors[0])
    print("Last color: ", colors[-1])
else:
    print("Please enter at least two color names.")
```

Q(13) Accept an integer n and compute n+nn+nnn

```
# Accept an integer from the user
n = int(input("Enter an integer: "))

# Compute nn and nnn
nn = int(str(n) * 2)  # Repeat n twice
nnn = int(str(n) * 3)  # Repeat n three times

# Compute n + nn + nnn
result = n + nn + nnn

# Print the result
print("Result: ", result)
```

Q(14) Print out all colors from color-list 1 not contained in color-list 2.

```
# Define color lists
```

```
color_list_1 = ['Red', 'Green', 'Blue', 'White',
'Orange']
color_list_2 = ['Green', 'Orange', 'Black', 'White']

# Print colors from color_list_1 not contained in
color_list_2
result_colors = [color for color in color_list_1 if
color not in color_list_2]
print("Colors from color_list_1 not contained in
color_list_2:", result_colors)
```

Q(15) Create a single string separated with space from two strings by swapping the character at position 1.

```
# Accept two strings from the user
string1 = input("Enter the first string: ")
string2 = input("Enter the second string: ")

# Check if both strings have at least one character
if len(string1) > 0 and len(string2) > 0:
    # Swap characters at position 1
    new_string1 = string2[0] + string1[1:]
    new_string2 = string1[0] + string2[1:]
    # Create a single string separated by space
    result_string = new_string1 + " " + new_string2
    # Print the result
    print("Resulting string: ", result_string)
else:
    print("Both strings must have at least one
character.")
```

Q(16) Sort dictionary in ascending and descending order.

```
# Dictionary

my_dict = {'one': 1, 'three': 3, 'five': 5, 'two': 2,
'four': 4}


# Sorting in ascending order by keys

sorted_dict_asc = dict(sorted(my_dict.items()))

print("Dictionary in Ascending Order by Keys:",
sorted_dict_asc)

# Sorting in descending order by keys

sorted_dict_desc = dict(sorted(my_dict.items(),
reverse=True))

print("Dictionary in Descending Order by Keys:",
sorted_dict_desc)


# Sorting in ascending order by values

sorted_dict_asc_values = dict(sorted(my_dict.items(),
key=lambda item: item[1]))

print("Dictionary in Ascending Order by Values:",
sorted_dict_asc_values)


# Sorting in descending order by values

sorted_dict_desc_values =
dict(sorted(my_dict.items(), key=lambda item:
item[1], reverse=True))

print("Dictionary in Descending Order by Values:",
sorted_dict_desc_values)
```

Q(17) Merge two dictionaries

```
# Two dictionaries to be merged

dict1 = {'a': 1, 'b': 2}

dict2 = {'b': 3, 'c': 4}


# Merge the dictionaries using the update() method

merged_dict = dict1.copy()  # Create a copy of the
first dictionary to preserve the original

merged_dict.update(dict2)    # Update the copy with
the second dictionary


print("Merged Dictionary (using update()):",
merged_dict)


# Two dictionaries to be merged

dict1 = {'a': 1, 'b': 2}

dict2 = {'b': 3, 'c': 4}


# Merge the dictionaries using dictionary unpacking

merged_dict = {**dict1, **dict2}


print("Merged Dictionary (using dictionary
unpacking):", merged_dict)
```

Q(18) Find GCD of two numbers.

```
def gcd(a, b):

    while b != 0:

        a, b = b, a % b

    return a
```

```
# Example usage
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

result = gcd(num1, num2)
print("GCD of", num1, "and", num2, "is", result)
```

Q(19) From a list of integers, create a list removing the even numbers.

```
# Original list of integers
original_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Create a new list removing the even numbers
odd_numbers_list = [num for num in original_list if
num % 2 != 0]

print("Original List:", original_list)
print("List with Odd Numbers Only:",
odd_numbers_list)
```

Q(20) To find the factorial of a number.

```
 # Function to calculate factorial using a loop
def factorial_loop(n):
    factorial = 1
    for i in range(1, n + 1):
        factorial *= i
    return factorial

# Example usage
```

```
num = int(input("Enter a number: "))
result = factorial_loop(num)
print("Factorial of", num, "is", result)

# Function to calculate factorial using recursion
def factorial_recursive(n):
    if n == 0:
        return 1
    else:
        return n * factorial_recursive(n - 1)

# Example usage
num = int(input("Enter a number: "))
result = factorial_recursive(num)
print("Factorial of", num, "is", result)
```