10.Implementing programs using Strings. (reverse, character count and replacing characters)

input:

```
# 1. Reverse a string
def reverse_string(s):
    return s[::-1]

# 2. Count characters
def count_characters(s):
    count = {}
    for char in s:
        if char != ' ':
            count[char] = count.get(char, 0) + 1
    return count

# 3. Replace characters
def replace_character(s, old, new):
    return s.replace(old, new)
```

```python
# Main program
def main():
    text = input("Enter a string: ")

    print("\n1. Reversed String:")
    print(reverse_string(text))

    print("\n2. Character Count:")
    counts = count_characters(text)
    for char, c in counts.items():
        print(f"'{char}': {c} times")

    print("\n3. Replace Characters:")
    old_char = input("Enter character to replace: ")
    new_char = input("Enter new character: ")
    replaced = replace_character(text, old_char, new_char)
    print("After replacement:", replaced)

# Run the program
main()
```

Output:

```
Enter a string: mahesh

1. Reversed String:
hseham

2. Character Count:
'm': 1 times
'a': 1 times
'h': 2 times
'e': 1 times
's': 1 times

3. Replace Characters:
Enter character to replace: mahesh
Enter new character: vaishnav
After replacement: vaishnav
```

11.Write a program to find if a given number is palindrome or not.

Input:

```python
# Function to check palindrome
def is_palindrome(number):
    original = str(number)
    reversed_num = original[::-1]
    return original == reversed_num

# Main program
num = int(input("Enter a number: "))

if is_palindrome(num):
    print(f"{num} is a Palindrome number.")
else:
    print(f"{num} is NOT a Palindrome number.")
```

Output:

```
Enter a number: 122334433221
122334433221 is a Palindrome number.
```

12.Create a The Tower of Hanoi classic problem in recursive programming. The problem consists of three rods and a number of disks of different sizes. The objective is to move all the disks from the first rod to the third rod, following these rules: a. Only one disk can be moved at a time. b. A disk can only be moved if it is the uppermost disk on a rod. No larger disk can be placed on top of a smaller disk.

Input:

```
# Function to solve Tower of Hanoi
def tower_of_hanoi(n, source, auxiliary, destination):
    if n == 1:
        print(f"Move disk 1 from {source} to {destination}")
        return
    # Move top n-1 disks from source to auxiliary
    tower_of_hanoi(n - 1, source, destination, auxiliary)
    # Move the nth disk to destination
    print(f"Move disk {n} from {source} to {destination}")
    # Move the n-1 disks from auxiliary to destination
    tower_of_hanoi(n - 1, auxiliary, source, destination)

# Main Program
n = int(input("Enter number of disks: "))
print("\nSteps to solve Tower of Hanoi:")
tower_of_hanoi(n, 'A', 'B', 'C')  # A = source, B = auxiliary, C
    = destination
```

Output:

```
Enter number of disks: 2

Steps to solve Tower of Hanoi:
Move disk 1 from A to B
Move disk 2 from A to C
Move disk 1 from B to C
```