NAME : VIBHUTI DESHMUKH

STUDENT ID : 23CY-21

COURSE : BSc. (Hons.) CHEMISTRY

SEMESTER : 2nd

COURSE NAME : PROGRAMMING USING PYTHON

Q.1. Write a program to determine the type of input using match statement.

```
def determine_type(input_data):
    match input_data:
        case int:
            print("Input is an integer")
        case float:
            print("Input is a float")
        case str:
            print("Input is a string")
        case list:
            print("Input is a list")
        case dict:
            print("Input is a dictionary")
        case tuple:
            print("Input is a tuple")
        case set:
            print("Input is a set")
```

```
        case bool:
            print("Input is a boolean")
        case _:
            print("Input is of an unknown type")

# Test the function with different types of input
determine_type(10)
determine_type(3.14)
determine_type("Hello")
determine_type([1, 2, 3])
determine_type({"a": 1, "b": 2})
determine_type((1, 2, 3))
determine_type({1, 2, 3})
determine_type(True)
```

Q.2. Find the factorial of a number using function.

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

# Example usage
number = 5
print("Factorial of", number, "is", factorial(number))
```

Q.3. Write a program to print even numbers using continue statement.

```
def print_even_numbers(n):
    print("Even numbers up to", n, "are:")
    for i in range(1, n + 1):
        if i % 2 != 0:  # If the number is odd, skip to the next iteration
            continue
        print(i)

# Test the function
print_even_numbers(10)
```

Q.4. Write a program to demonstrate list, set, tuple and dictionary.

```
# List
fruits = ['apple', 'banana', 'orange', 'apple']
print("List:")
print("Fruits:", fruits)
print("Length:", len(fruits))
print("First fruit:", fruits[0])
print("Last fruit:", fruits[-1])
print("Index of 'banana':", fruits.index('banana'))

# Set
unique_fruits = set(fruits)
print("\nSet:")
print("Unique fruits:", unique_fruits)
unique_fruits.add('grape')
print("After adding 'grape':", unique_fruits)
```

```
# Tuple
person = ('John', 30, 'Male')
print("\nTuple:")
print("Name:", person[0])
print("Age:", person[1])
print("Gender:", person[2])

# Dictionary
student = {'name': 'Alice', 'age': 25, 'major': 'Computer Science'}
print("\nDictionary:")
print("Student:", student)
print("Name:", student['name'])
print("Age:", student['age'])
print("Major:", student['major'])
student['age'] = 26  # Update age
print("Updated age:", student['age'])
print("Keys:", student.keys())
print("Values:", student.values())
```

Q.5. Write a program to count number of vowels in a string.

```
def count_vowels(string):
    # Define a set of vowels
    vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}

    # Initialize a counter for vowels
    vowel_count = 0
    # Iterate through each character in the string
    for char in string:
        # Check if the character is a vowel
        if char in vowels:
            # Increment the vowel count
            vowel_count += 1

    # Return the total vowel count
    return vowel_count
# Test the function
input_string = input("Enter a string: ")
print("Number of vowels in the string:", count_vowels(input_string))
```

Q.6. Write a program to find maximum and minimum of n numbers.

```
def find_max_min(numbers):
    if not numbers:
        return None, None

    max_num = numbers[0]
    min_num = numbers[0]

    for num in numbers:
        if num > max_num:
            max_num = num
        elif num < min_num:
            min_num = num

    return max_num, min_num

def main():
    n = int(input("Enter the number of elements: "))
    numbers = []
```

```
for i in range(n):
        num = float(input(f"Enter number {i + 1}: "))
        numbers.append(num)

    max_num, min_num = find_max_min(numbers)

    if max_num is not None and min_num is not None:
        print(f"The maximum number is: {max_num}")
        print(f"The minimum number is: {min_num}")
    else:
        print("No numbers were entered.")

if __name__ == "__main__":
    main()
```

Q.7. Write a program to check number is prime or not.

```
def is_prime(num):
    if num <= 1:
        return False
    elif num <= 3:
        return True
    elif num % 2 == 0 or num % 3 == 0:
        return False
    i = 5
    while i * i <= num:
        if num % i == 0 or num % (i + 2) == 0:
            return False
        i += 6
    return True
# Test the function
num = int(input("Enter a number: "))
if is_prime(num):
    print(num, "is a prime number.")
else:
    print(num, "is not a prime number.")
```

Q.8. Write a program to display the first occurrence of number divisible by ‘k’ in the list.

```
def first_divisible_number(numbers, k):
    for num in numbers:
        if num % k == 0:
            return num
    return None

# Example usage:
numbers_list = [2, 3, 7, 10, 12, 15]
k = 5

result = first_divisible_number(numbers_list, k)
if result is not None:
    print(f"The first number divisible by {k} is: {result}")
else:
    print(f"There is no number divisible by {k} in the list.")
```

Q.9. Write a program to count occurrence of each element in the list.

```
def count_occurrences(lst):
    # Create an empty dictionary to store element counts
    counts = {}
    # Iterate through each element in the list
    for item in lst:
        # Check if the element is already in the dictionary
        if item in counts:
            # If it is, increment its count
            counts[item] += 1
        else:
            # If it's not, add it to the dictionary with a count of 1
            counts[item] = 1
    return counts
# Example usage:
my_list = [1, 2, 3, 4, 1, 2, 3, 1, 2, 1]
result = count_occurrences(my_list)
print("Occurrences of each element in the list:")
for item, count in result.items():
    print(f"{item}: {count}")
```

Q.10. Check a string is palindrome or not.

```
def is_palindrome(s):
    # Convert the string to lowercase and remove non-alphanumeric characters
    s = ''.join(char.lower() for char in s if char.isalnum())

    # Check if the string is equal to its reverse
    return s == s[::-1]


# Test the function
string = input("Enter a string: ")
if is_palindrome(string):
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

Q.11. Generate Fibonacci sequence upto n terms using while loop.

```
def fibonacci(n):
    fib_sequence = []
    a, b = 0, 1
    count = 0
    while count < n:
        fib_sequence.append(a)
        a, b = b, a + b
        count += 1
    return fib_sequence

# Example: Generate Fibonacci sequence up to 10 terms
n = 10
fib_sequence = fibonacci(n)
print("Fibonacci sequence up to", n, "terms:", fib_sequence)
```

Q.12. Write a program to find all prime numbers from 1 to n numbers.

```
def is_prime(num):
    if num <= 1:
        return False
    if num <= 3:
        return True
    if num % 2 == 0 or num % 3 == 0:
        return False
    i = 5
    while i * i <= num:
        if num % i == 0 or num % (i + 2) == 0:
            return False
        i += 6
    return True
```

```
def find_primes(n):
    primes = []
    for i in range(2, n + 1):
        if is_prime(i):
            primes.append(i)
    return primes

n = int(input("Enter the value of n: "))
print("Prime numbers from 1 to", n, "are:")
print(find_primes(n))
```

Q.13. Write a program to print all the names from list whose length greater than 6.

```
def print_names_longer_than_six(names):
    for name in names:
        if len(name) > 6:
            print(name)

# Example list of names
names_list = ["Jonathan", "Elizabeth", "Christopher", "Emma", "Alexander", "Isabella", "Michael", "Sophia"]

# Call the function with the list of names
print_names_longer_than_six(names_list)
```

Q.14. Find the sum of digit of a number.

```
def sum_of_digits(number):
    # Initialize the sum
    sum = 0

    # Loop through each digit of the number
    while number > 0:
        # Extract the last digit of the number
        digit = number % 10
        # Add the extracted digit to the sum
        sum += digit
        # Remove the last digit from the number
        number //= 10

    return sum

# Example usage
number = int(input("Enter a number: "))
print("Sum of digits:", sum_of_digits(number))
```

Q.15. Write a program to check a number is binary.

```
def is_binary(number):
    for digit in str(number):
        if digit != '0' and digit != '1':
            return False
    return True

# Test the function
number = input("Enter a number: ")

if is_binary(number):
    print(number, "is a binary number.")
else:
    print(number, "is not a binary number.")
```

Q.16. Write a program to remove vowels from string.

```
def remove_vowels(string):
    vowels = "aeiouAEIOU"
    without_vowels = ""
    for char in string:
        if char not in vowels:
            without_vowels += char
    return without_vowels

# Example usage:
input_string = input("Enter a string: ")
print("String without vowels:", remove_vowels(input_string))
```

Q.17. Write a program to display nth Fibonacci number.

```
def fibonacci(n):
    if n <= 0:
        return "Invalid input. Please provide a positive integer."
    elif n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)

# Test the function
n = int(input("Enter the value of n: "))
print("The", n, "th Fibonacci number is:", fibonacci(n))
```

Q.18. Check a number in Armstrong.

```
def is_armstrong(num):
    # Count the number of digits
    num_str = str(num)
    num_digits = len(num_str)

    # Calculate the sum of digits raised to the power of the number of digits
    armstrong_sum = sum(int(digit) ** num_digits for digit in num_str)
    # Check if the sum is equal to the original number
    return armstrong_sum == num


# Test the function
number = int(input("Enter a number to check if it's an Armstrong number: "))
if is_armstrong(number):
    print(number, "is an Armstrong number.")
else:
    print(number, "is not an Armstrong number.")
```

Q.19. Write a program to print ASCII value of all the characters of string with character.

```
def print_ascii(string):
    print("Character\tASCII Value")
    for char in string:
        ascii_value = ord(char)
        print(f"{char}\t\t{ascii_value}")

# Example usage:
input_string = input("Enter a string: ")
print_ascii(input_string)
```

Q.20. Write a program to check whether a list is monoatomic or not.

```
def is_monoatomic(lst):
    if len(lst) == 0:
        return False
    element = lst[0]
    for item in lst:
        if item != element:
            return False
    return True


# Example usage
sample_list = [2, 2, 2, 2]
if is_monoatomic(sample_list):
    print("The list is monoatomic.")
else:
    print("The list is not monoatomic.")
```

Q.21. Write a program to check a particular element is present in the array or not.

```
def check_element(arr, element):
    for item in arr:
        if item == element:
            return True
    return False

# Example usage:
arr = [1, 2, 3, 4, 5]
element_to_check = 3

if check_element(arr, element_to_check):
    print(f"The element {element_to_check} is present in the array.")
else:
    print(f"The element {element_to_check} is not present in the array.")
```

Q.22. Write a program to find nth largest and nth smallest element from list.

```
def nth_largest_smallest(nums, n):
    nums.sort()
    nth_largest = nums[-n]
    nth_smallest = nums[n - 1]
    return nth_largest, nth_smallest


# Example usage:
if __name__ == "__main__":
    lst = [4, 7, 1, 9, 3, 5, 8]
    n = 3
    largest, smallest = nth_largest_smallest(lst, n)
    print(f"{n}th largest element: {largest}")
    print(f"{n}th smallest element: {smallest}")
```

Q.23. Write a program to remove kth character from string.

```
def remove_kth_character(input_string, k):
    if k < 0 or k >= len(input_string):
        return "Invalid value of k"

    return input_string[:k] + input_string[k+1:]

# Example usage:
input_string = "example"
k = 2
result = remove_kth_character(input_string, k)
print("Result:", result)
```

Q.24. Check a particular substring present in the string.

```
def check_substring(string, substring):
    if substring in string:
        return True
    else:
        return False

# Example usage:
string = "This is a sample program."
substring = "program"
if check_substring(string, substring):
    print(f"The substring '{substring}' is present in the string.")
else:
    print(f"The substring '{substring}' is not present in the string.")
```

Q.25. Write a program to print all the characters whose length is even.

```
def print_even_length_characters(string):
    for char in string:
        if len(char) % 2 == 0:
            print(char)

# Example usage:
input_string = "Hello World!"
print_even_length_characters(input_string)
```

Q.26. Remove duplicate elements from list.

```
def remove_duplicates(input_list):
    unique_list = []
    for item in input_list:
        if item not in unique_list:
            unique_list.append(item)
    return unique_list

def main():
    my_list = [1, 2, 3, 4, 2, 3, 5]
    result = remove_duplicates(my_list)
    print("Original List:", my_list)
    print("List without duplicates:", result)

if __name__ == "__main__":
    main()
```

Q.27. Write a program to display a character whose is more than other characters.

```
def most_common_character(input_string):
    # Dictionary to store character frequencies
    char_frequency = {}

    # Count frequencies of characters
    for char in input_string:
        if char in char_frequency:
            char_frequency[char] += 1
        else:
            char_frequency[char] = 1

    # Find the character with the maximum frequency
    max_char = ''
    max_count = 0
    for char, count in char_frequency.items():
```

```
    if count > max_count:
            max_char = char
            max_count = count

    return max_char

# Input string
input_string = input("Enter a string: ")

# Call the function and display the most common character
most_common_char = most_common_character(input_string)
print(f"The most common character in the string '{input_string}' is '{most_common_char}'")
```

Q.28. Find words whose length is greater than j and less than k.

```
def find_words_with_length_between(words, j, k):
    result = []
    for word in words:
        if j < len(word) < k:
            result.append(word)
    return result

# Example usage:
words = ["apple", "banana", "orange", "grape", "kiwi", "pineapple"]
j = 4
k = 7
result = find_words_with_length_between(words, j, k)
print("Words with length between", j, "and", k, ":", result)
```

Q.29. Check a string is binary or not.

```
def is_binary_string(s):
    for char in s:
        if char != '0' and char != '1':
            return False
    return True

# Example usage:
input_string = "101010101"
if is_binary_string(input_string):
    print("The string is binary.")
else:
    print("The string is not binary.")
```

Q.30. Find uncommon words from two string.

```
def find_uncommon_words(string1, string2):
    # Tokenize strings into words
    words1 = string1.split()
    words2 = string2.split()
    # Create sets of unique words
    set1 = set(words1)
    set2 = set(words2)

    # Find uncommon words using symmetric difference
    uncommon_words = set1.symmetric_difference(set2)

    return uncommon_words
# Example usage:
string1 = "This is a sample string."
string2 = "This is another sample string with some different words."
uncommon_words = find_uncommon_words(string1, string2)
print("Uncommon words:", uncommon_words)
```

Q.31. Check a mail is valid or not.

```
import re
import dns.resolver


def validate_email(email):
    # Regular expression pattern for email validation
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

    if re.match(pattern, email):
        # Split email address to extract domain
        _, domain = email.split('@')

        try:
            # Query DNS records to check domain validity
            dns.resolver.resolve(domain, 'MX')
```

```
return True
        except dns.resolver.NoAnswer:
            print("No MX record found for the domain. Email might be invalid.")
            return False
        except dns.resolver.NXDOMAIN:
            print("Domain does not exist. Email might be invalid.")
            return False
    else:
        print("Invalid email format.")
        return False

# Example usage
email_address = input("Enter an email address to validate: ")
if validate_email(email_address):
    print("Email address is valid.")
else:
    print("Email address is not valid.")
```

Q.32. Check a mobile number is valid or not.

```
import re
def is_valid_mobile_number(number):
    # Regular expression to match common mobile number formats
    pattern = re.compile(r'^(\+\d{1,3})?[-.\s]?\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}$')
    # Check if the number matches the pattern
    if re.match(pattern, number):
        return True
    else:
        return False

def main():
    mobile_number = input("Enter the mobile number to check its validity: ")
    if is_valid_mobile_number(mobile_number):
        print("The mobile number is valid.")
    else:
        print("The mobile number is not valid.")

if __name__ == "__main__":
    main()
```

Q.33. Display only digits from tuple.

```
# Sample tuple
my_tuple = (123, 'abc', 456, 'def', 789)

# List to store digits
digits_list = []

# Iterate over the tuple
for item in my_tuple:
    if isinstance(item, str):  # Check if item is a string
        for char in item:
            if char.isdigit():  # Check if character is a digit
                digits_list.append(char)
    elif isinstance(item, int):  # Check if item is an integer
        digits_list.extend(str(item))

# Convert list of digits to string
digits_str = ''.join(digits_list)

print(digits_str)
```

Q.34. Write a program to display a character having higher occurrence.

```
def most_common_character(string):
    # Create a dictionary to store character counts
    char_count = {}

    # Count occurrences of each character in the string
    for char in string:
        if char in char_count:
            char_count[char] += 1
        else:
            char_count[char] = 1

    # Find the maximum occurrence
    max_count = max(char_count.values())
```

```
    # Find the characters with maximum occurrence
    most_common_chars = [char for char, count in char_count.items() if count == max_count]

    return most_common_chars

# Example usage
input_string = input("Enter a string: ")
result = most_common_character(input_string)

print("Character(s) with the highest occurrence:", ", ".join(result))
```