Lab-2

1.
```
def count_vowels_and_consonants(input_string):
    # Define vowels and initialize counters
    vowels = 'aeiouAEIOU'
    vowel_count = 0
    consonant_count = 0
   # Loop through each character in the input string
    for char in input_string:
        # Check if the character is a letter
        if char.isalpha():
            if char in vowels:
                vowel_count += 1
            else:

        consonant_count += 1

     return vowel_count, consonant_count

# Get input from the user

  user_input = input("Enter a string: ")

# Count vowels and consonants

  vowels, consonants = count_vowels_and_consonants(user_input)

print(f"Number of vowels: {vowels}")

print(f"Number of consonants: {consonants}")
```

2.
```
def get_matrix_input(name):
    """ Function to get matrix input from the user """
    rows = int(input(f"Enter the number of rows for matrix {name}: "))

   cols = int(input(f"Enter the number of columns for matrix {name}: "))

 matrix = []

 print(f"Enter the elements of matrix {name} row by row:")

 for i in range(rows):

   row = list(map(int, input(f"Row {i + 1}: ").split()))

    if len(row) != cols:

        raise ValueError(f"Row does not have the correct number of columns. Expected {cols}, got
{len(row)}.")

    matrix.append(row)

 return matrix, rows, cols

def multiply_matrices(A, B, rowsA, colsA, rowsB, colsB):
```

```
""" Function to multiply two matrices A and B """
        if colsA != rowsB:
            return "Error: The number of columns in matrix A must equal the number of rows in matrix
    B."
        # Initialize the result matrix with zeros
        result = [[0] * colsB for _ in range(rowsA)]
        # Perform matrix multiplication
        for i in range(rowsA):
            for j in range(colsB):
                result[i][j] = sum(A[i][k] * B[k][j] for k in range(colsA))
         return result
    def print_matrix(matrix):
        """ Function to print a matrix """
        for row in matrix:
            print(" ".join(map(str, row)))
    # Main program
    try:
        # Get matrices from the user
        A, rowsA, colsA = get_matrix_input("A")
        B, rowsB, colsB = get_matrix_input("B")
        # Multiply matrices and get the result
        result = multiply_matrices(A, B, rowsA, colsA, rowsB, colsB)
        # Print result
        if isinstance(result, str):  # If result is an error message
            print(result)
        else:
            print("Product of matrices A and B is:")
           print_matrix(result)
    except ValueError as e:
        print(f"Input error: {e}")
```

3. def count_common_elements(list1, list2):

```
    """ Function to count the number of common elements between two lists """
    # Convert lists to sets
    set1 = set(list1)
    set2 = set(list2)
     # Find intersection
    common_elements = set1.intersection(set2)
     # Return the number of common elements
    return len(common_elements)
def get_list_input(name):
    """ Function to get a list of integers from the user """
    return list(map(int, input(f"Enter integers for list {name} (separated by spaces): ").split()))


  # Main program
  # Get input lists from the user
  list1 = get_list_input("A")
  list2 = get_list_input("B")
 # Find the number of common elements
  common_count = count_common_elements(list1, list2)
 # Print the result
  print(f"The number of common elements between the two lists is: {common_count}")
```

4. def get_matrix_input():

```
    """Function to get a matrix input from the user."""
    rows = int(input("Enter the number of rows: "))
    cols = int(input("Enter the number of columns: "))
     matrix = []
    print("Enter the elements of the matrix row by row:")
    for i in range(rows):
        row = list(map(int, input(f"Row {i + 1}: ").split()))
        if len(row) != cols:
            raise ValueError(f"Row does not have the correct number of columns. Expected {cols}, got
   {len(row)}.")
          matrix.append(row)
 return matrix, rows, cols
def transpose_matrix(matrix, rows, cols):
      """Function to compute the transpose of a matrix."""
      # Initialize the transpose matrix with the reversed dimensions
      transpose = [[0] * rows for _ in range(cols)]
       # Compute transpose
      for i in range(rows):
         for j in range(cols):
            transpose[j][i] = matrix[i][j]
      return transpose
   def print_matrix(matrix):
      """Function to print a matrix."""
      for row in matrix:
         print(" ".join(map(str, row)))


   # Main program
   try:
      # Get matrix input from the user
      matrix, rows, cols = get_matrix_input()
      # Compute the transpose
      transposed = transpose_matrix(matrix, rows, cols)
      # Print the transposed matrix
      print("Transpose of the matrix is:")
      print_matrix(transposed)
   except ValueError as e:
      print(f"Input error: {e}")
```