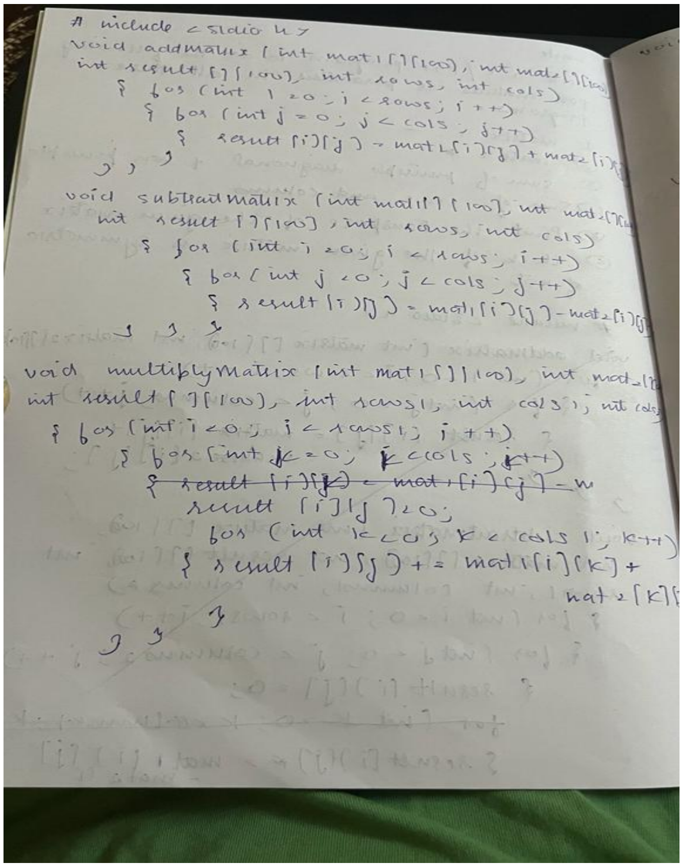
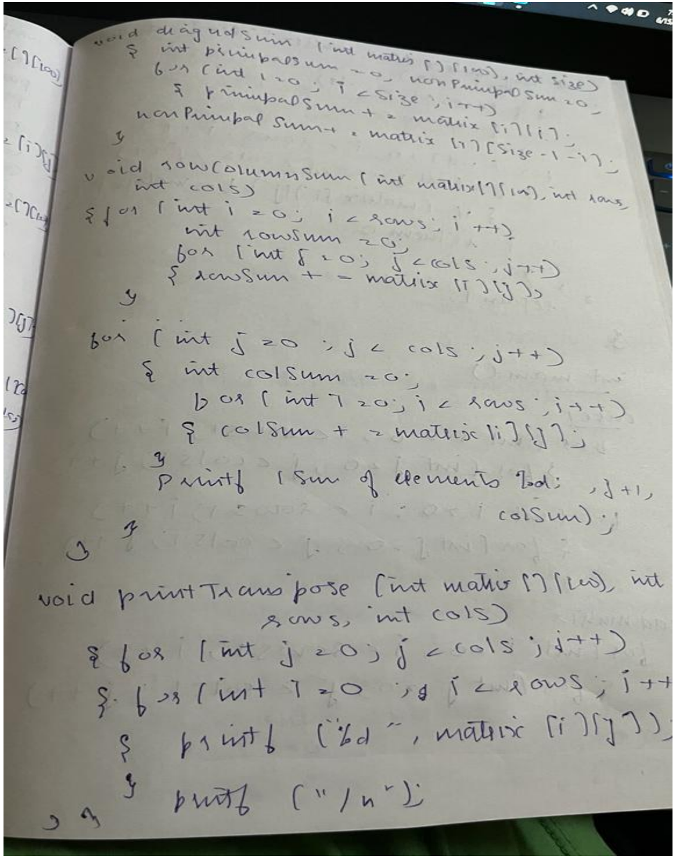
WRITE A C\C++ PROGRAM TO DO THE FOLLOWING . PASS MATRICES AS PARAMETERS IN ALL THESE PROGRAMS.

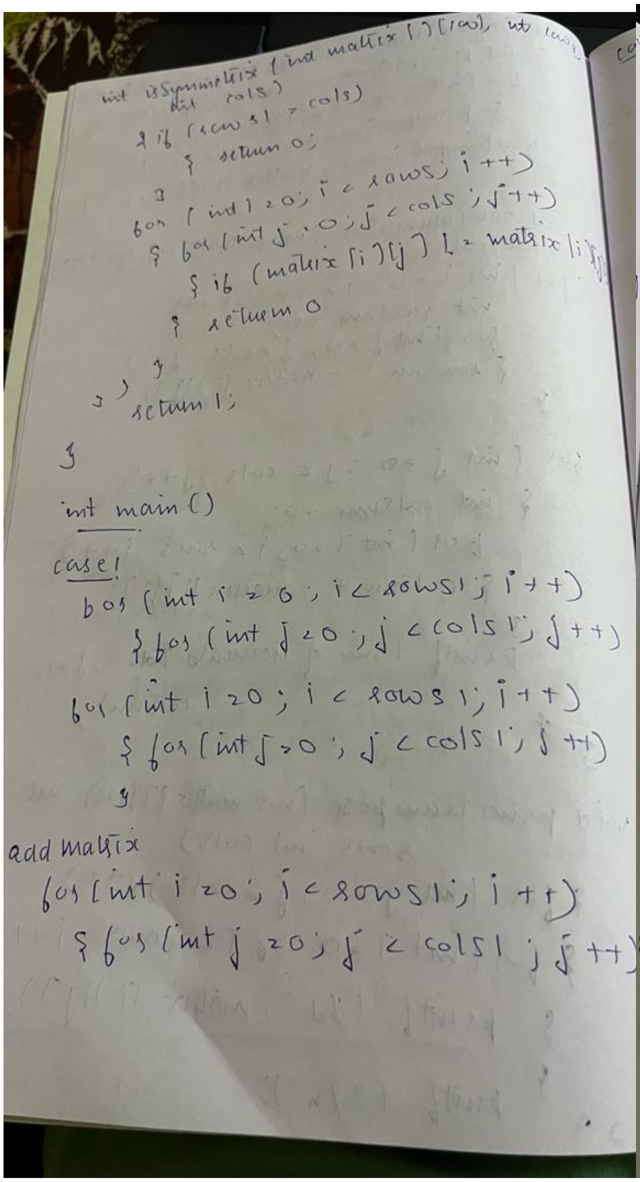
1) MATRIX +,-
2) MATRIX *
3) SUM OF PRINCIPAL DIAGONAL AND NON-PRINCIPAL
4) SUM OF ROWS AND COLUMNS
5) PRINT THE TRANSPOSE OF THE GIVEN MATRIX
6) CHECK IF THE GIVEN MATRIX IS SYMMETRIC OR NOT.



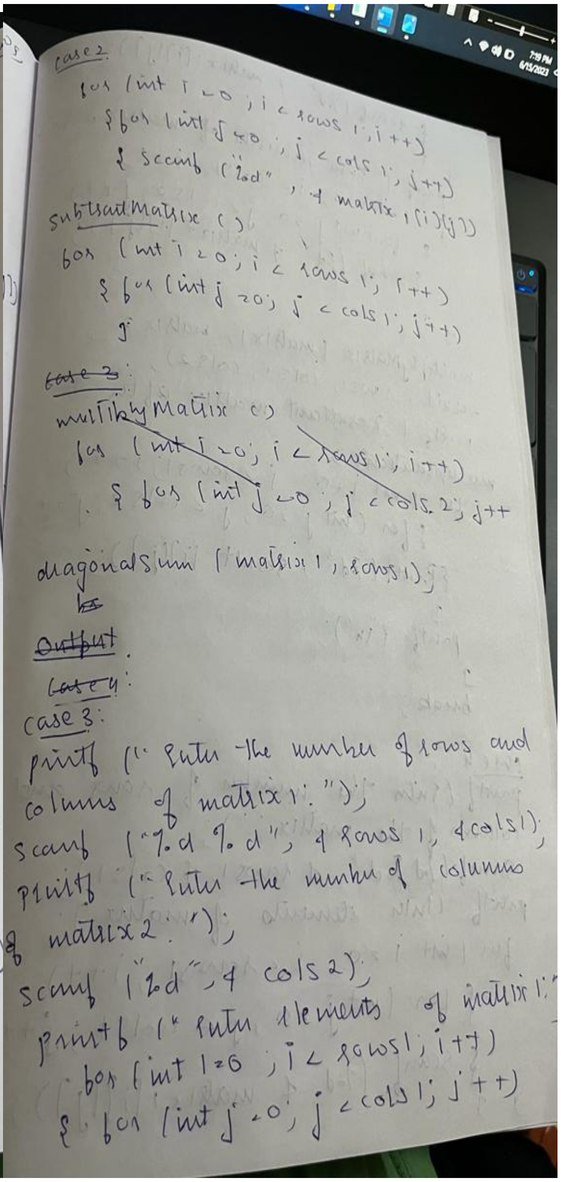
```
# include <stdio h>
void addMatrix (int mat1[][100], int mat2[][100]
int result [][100], int rows, int cols)
    { for (int i =0 ; i < rows; i ++)
        { for (int j = 0 ; j < cols ; j++)
            {   result [i][j] = mat1[i][j] + mat2 [i][j]
  } }    }

void subtractMatrix (int mat1[] [100], int mat2[][100]
    int result [][100] , int rows, int cols)
        { for (int i =0; i < rows; i++)
            { for (int j =0 ; j < cols ; j++)
                { result [i][j] = mat1[i][j] - mat2 [i][j]
        }  }  }

void multiplyMatrix (int mat1 []][100], int mat2[]
int result [ ][100], int rows1, int cols 1, int cols
  { for (int i =0 ; i < rows1; i ++)
      { for (int k =0; k <cols ; k++)
        ~~{ result [i][k] = mat1[i][j] - m~~
          result [i][j] =0;
            for (int k =0; k < cols 1; k++)
          { result [i][j] += mat1[i][k] +
                                  mat 2 [k][
            }
    }  }
```



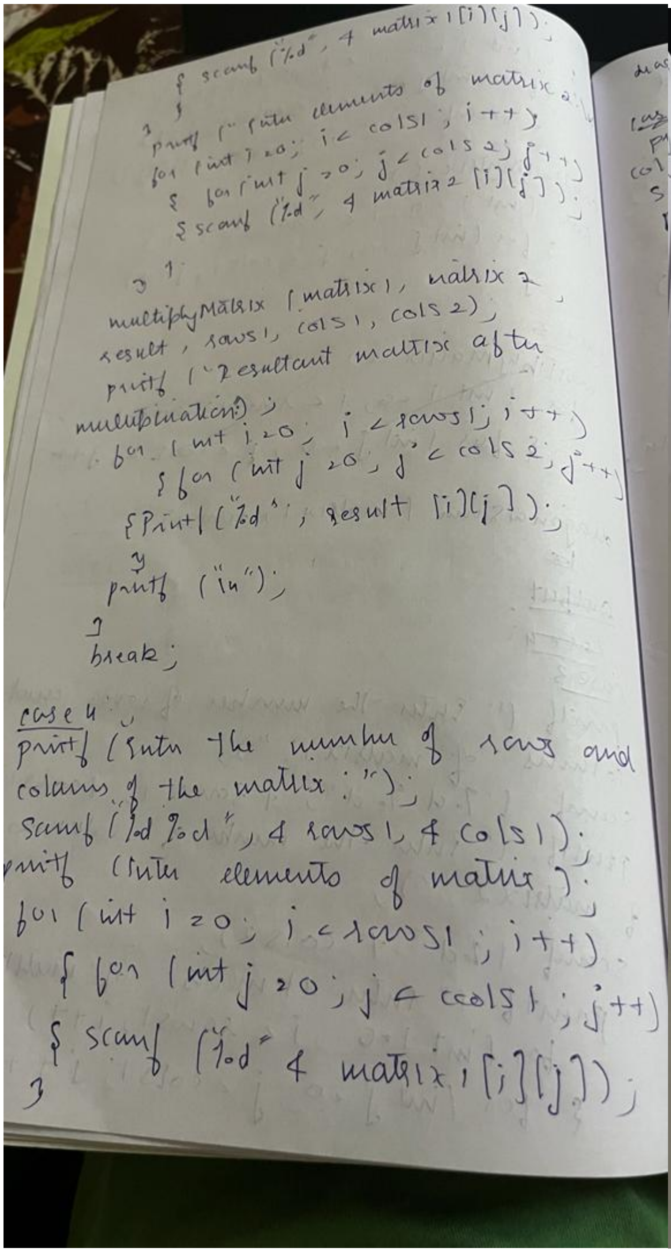
```
void diagnalSum (int matrix [][100], int size)
  { int principalSum =0, nonPrincipalSum =0,
    for (int i =0 ; i <size ; i++)
      { principalSum + = matrix [i][i];
  nonPrincipal Sum+ = matrix [i][size - 1 - i];
  }

void rowColumnSum (int matrix[][100], int rows,
  int cols)
{ for (int i =0; i < rows; i ++)
      int rowSum =0;
    for (int j =0; j <cols ; j++)
    { rowSum + = matrix [i][j];
  }

for (int j =0 ; j < cols ; j++)
    { int colSum =0;
       for (int i =0; i < rows ; i++)
      { colSum + = matrix [i][j];
    }
     printf (Sum of elements %d: , j+1,
                                  colSum);
    }
 }

void printTranspose (int matrix [][100], int
                rows, int cols)
  { for (int j =0; j < cols ; j++)
   { for (int i =0 ; i < rows ; i++
      {  printf ("%d ", matrix [i][j]);
      }  printf ("/n");
 } }
```



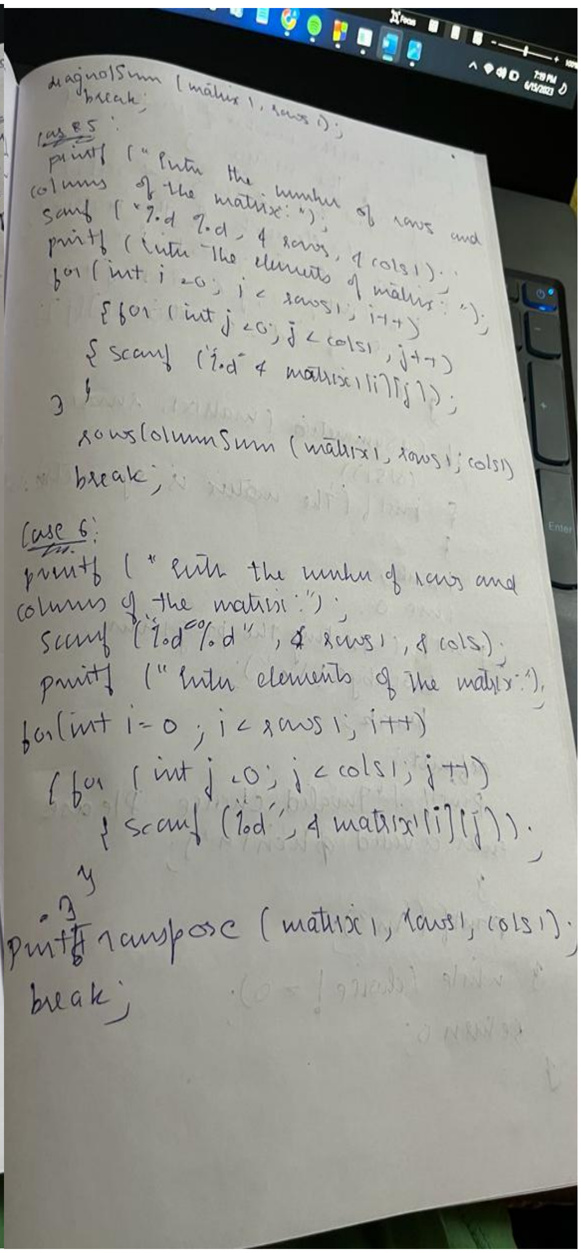
```
int isSymmetric (int matrix [ ][100], int rows,
     int cols)
  { if (rows1 = cols)
      { return 0;
  }
  for ( int i = 0; i < rows; i ++)
  { for (int j = 0; j < cols ; j ++)
      { if (matrix [i][j] != matrix [j][i])
          { return 0
      }
    }
  } return 1;
}

int main ()

case1
  for (int i = 0 ; i < rows1; i ++)
     { for (int j = 0 ; j < cols1; j ++)

  for (int i = 0 ; i < rows 1; i ++)
     { for (int j = 0 ; j < cols 1; j ++)
     }

add matrix
  for (int i = 0; i < rows1; i ++)
    { for (int j = 0; j < cols1 ; j ++)}
```



```
case2.
  for (int i = 0 ; i < rows 1; i ++)
    {for (int j = 0 ; j < cols 1; j ++)
      { scanf ("%d" , & matrix1[i][j])
subtractMatrix ( )
for (int i = 0; i < rows 1; i ++)
    { for (int j = 0; j < cols 1; j ++)
      }

~~case 3~~:
multiplyMatrix ( )
  for (int i = 0; i < rows 1; i ++)
    { for (int j = 0 ; j < cols 2; j++

diagonalSum (matrix1, rows1);
   ~~ts~~
~~Output~~.
 ~~case 4~~:
case 3:
printf ("Enter the number of rows and
columns of matrix1: ");
scanf ("%d %d", & rows 1, & cols1);
printf ("Enter the number of columns
of matrix2: ");
scanf ("%d", & cols2);
 printf ("Enter elements of matrix1:"
    for (int i = 0 ; i < rows1; i ++)
  { for (int j = 0; j < cols1; j ++)
```

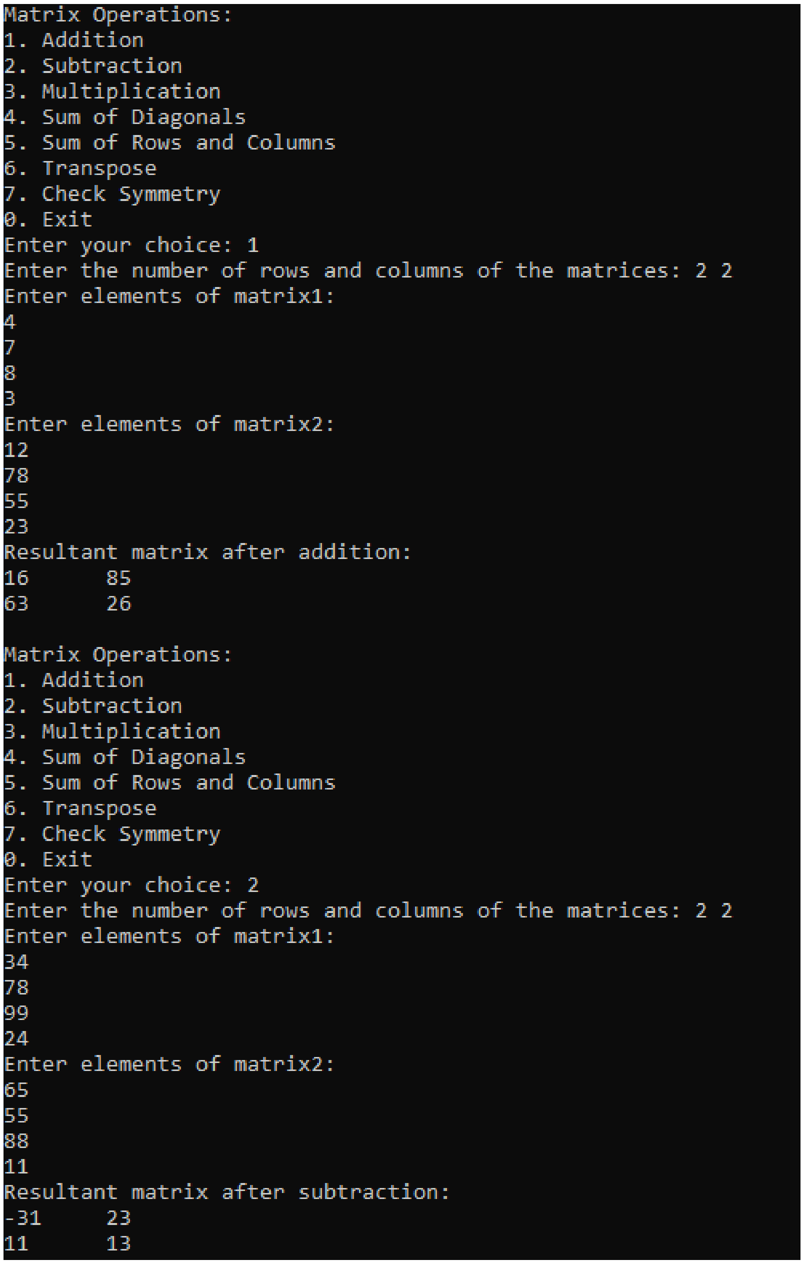


```
{ scanf ("%d", & matrix1[i][j]);
}
}
printf ("Enter elements of matrix 2
for (int i=0; i< cols1; i++)
{ for (int j=0; j<cols2; j++)
{ scanf ("%d", & matrix2 [i][j]);
}
}
multiplyMatrix (matrix1, matrix2,
result, rows1, cols1, cols2);
printf ("Resultant matrix after
multiplication:");
for (int i=0; i<rows1; i++)
{ for (int j=0; j<cols2; j++)
{ Print ("%d ", result [i][j]);
}
printf ("\n");
}
break;

case 4:
printf (Enter the number of rows and
columns of the matrix :");
scanf (%d %d", & rows1, & cols1);
printf (Enter elements of matrix ");
for (int i=0; i<rows1; i++)
{ for (int j=0; j<ccols1; j++)
{ scanf ("%d" & matrix1[i][j]);
}
```

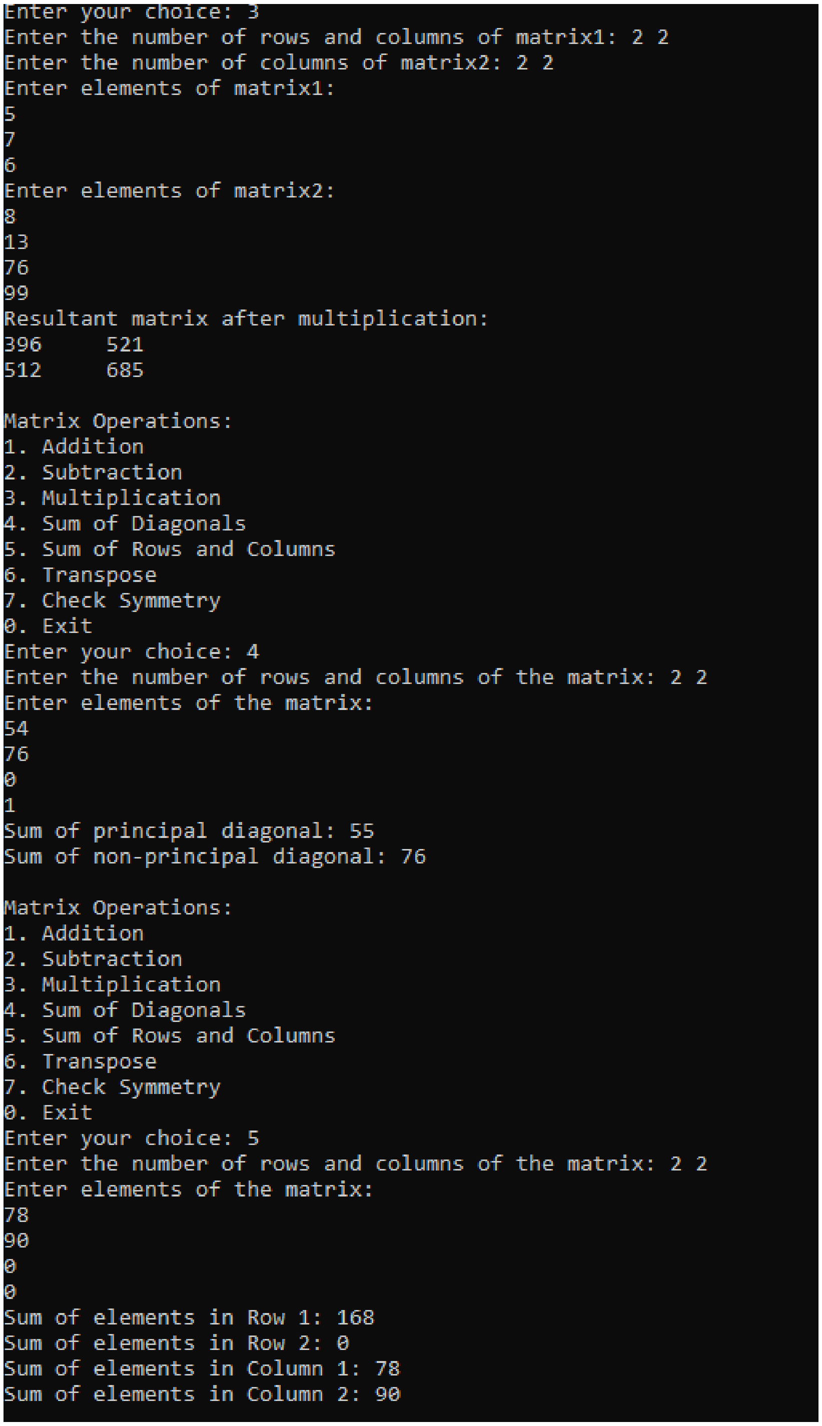


```
diagnolSum (matrix1, rows1);
break;
case 5:
printf ("Enter the number of rows and
columns of the matrix:");
scanf ("%d %d", & rows, & cols1);
printf (Enter the elements of matrix:");
for (int i=0; i< rows1; i++)
{ for (int j=0; j<cols1, j++)
{ scanf ("%d" & matrix1[i][j]);
}
}
rowsColumnSum (matrix1, rows1, cols1)
break;

case 6:
printf ("Enter the number of rows and
columns of the matrix:");
scanf ("%d%d", & rows1, & cols);
printf ("Enter elements of the matrix:");
for(int i=0; i<rows1; i++)
{ for (int j=0; j<cols1; j++)
{ scanf ("%d", & matrix1[i][j]);
}
}
printTranspose (matrix1, rows1, cols1);
break;
```

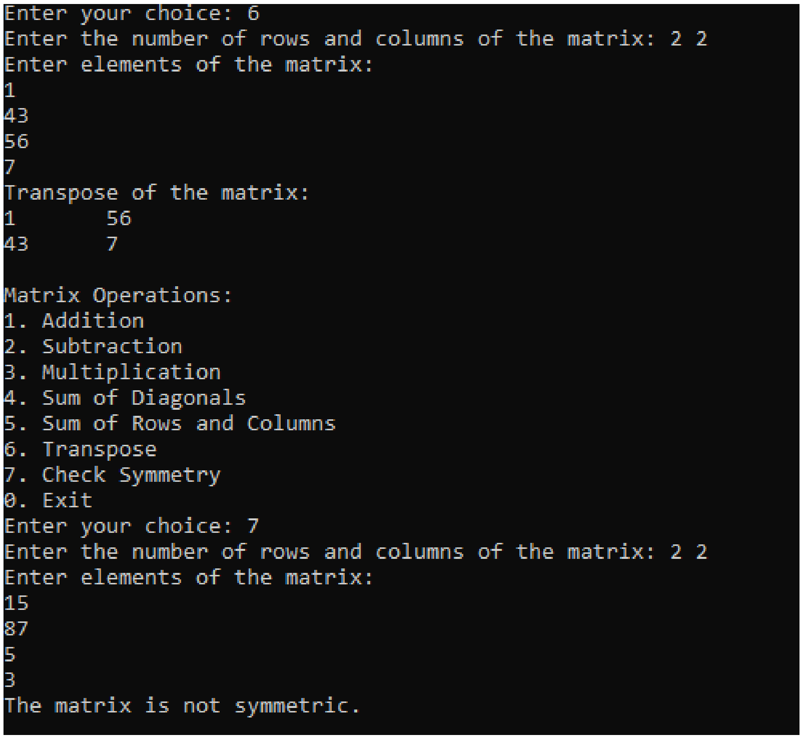
```
diagnolSum (matrix1, rows1);
    break;
case 5:
  printf ("Enter the number of rows and
columns of the matrix: ");
  scanf ("%d %d, & rows, & cols1).
  printf (Enter the elements of matrix: ");
  for (int i =0; i < rows1; i++)
     {for (int j =0; j < cols1 , j++)
    { scanf ("%d" & matrix1[i][j]);
    }
  }
    rowsColumnSum (matrix1, rows1, cols1)
    break;

case 6:
printf (" Enter the number of rows and
columns of the matrix: ");
  scanf ("%d"%d", & rows1, & cols);
  printf ("Enter elements of the matrix:
for(int i=0 ; i< rows1; i++)
  { for (int j =0; j < cols1; j++)
      { scanf ("%d", & matrix1[i][j]).
    }
  }
printTranspose (matrix1, rows1, cols1
 break;

   }
```



```
Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of Diagonals
5. Sum of Rows and Columns
6. Transpose
7. Check Symmetry
0. Exit
Enter your choice: 1
Enter the number of rows and columns of the matrices: 2 2
Enter elements of matrix1:
4
7
8
3
Enter elements of matrix2:
12
78
55
23
Resultant matrix after addition:
16      85
63      26

Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of Diagonals
5. Sum of Rows and Columns
6. Transpose
7. Check Symmetry
0. Exit
Enter your choice: 2
Enter the number of rows and columns of the matrices: 2 2
Enter elements of matrix1:
34
78
99
24
Enter elements of matrix2:
65
55
88
11
Resultant matrix after subtraction:
-31     23
11      13
```



```
Enter your choice: 3
Enter the number of rows and columns of matrix1: 2 2
Enter the number of columns of matrix2: 2 2
Enter elements of matrix1:
5
7
6
Enter elements of matrix2:
8
13
76
99
Resultant matrix after multiplication:
396     521
512     685

Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of Diagonals
5. Sum of Rows and Columns
6. Transpose
7. Check Symmetry
0. Exit
Enter your choice: 4
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
54
76
0
1
Sum of principal diagonal: 55
Sum of non-principal diagonal: 76

Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of Diagonals
5. Sum of Rows and Columns
6. Transpose
7. Check Symmetry
0. Exit
Enter your choice: 5
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
78
90
0
0
Sum of elements in Row 1: 168
Sum of elements in Row 2: 0
Sum of elements in Column 1: 78
Sum of elements in Column 2: 90
```



```
Enter your choice: 6
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
1
43
56
7
Transpose of the matrix:
1       56
43      7

Matrix Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Sum of Diagonals
5. Sum of Rows and Columns
6. Transpose
7. Check Symmetry
0. Exit
Enter your choice: 7
Enter the number of rows and columns of the matrix: 2 2
Enter elements of the matrix:
15
87
5
3
The matrix is not symmetric.
```