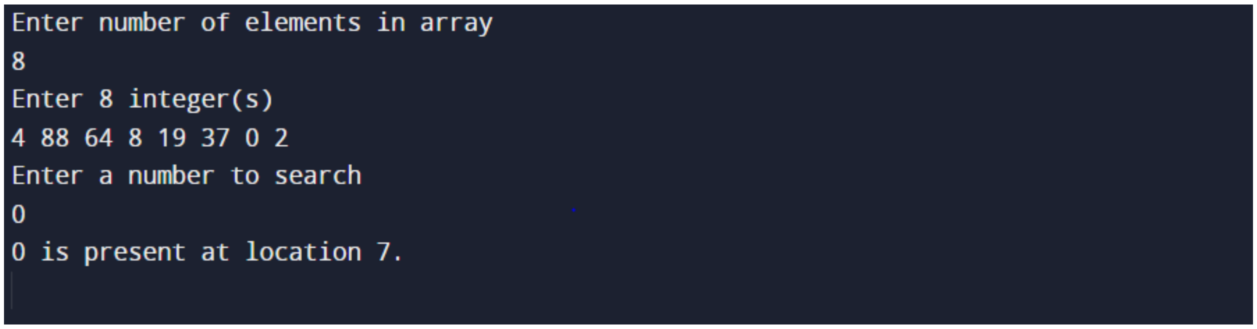
NAME: Yash Sarang
ROLL NO.: 47
CLASS: D1AD

CP LAB ASSIGNMENT 4

SECTION 1:
THEORY:
An array is defined as the collection of similar types of data items stored at contiguous memory locations. Arrays are the derived data type in C programming language which can store the primitive type of data such as int, char, double, float, etc. It also has the capability to store the collection of derived data types, such as pointers, structure, etc. The array is the simplest data structure where each data element can be randomly accessed by using its index number.
The array contains the following properties.

- Each element of an array is of the same data type and carries the same size, i.e., int = 4 bytes.
- Elements of the array are stored at contiguous memory locations where the first element is stored at the smallest memory location.
- Elements of the array can be randomly accessed since we can calculate the address of each element of the array with the given base address and the size of the data element.

Syntax: data_type array_name[array_size];

(**a**)
AIM: To implement Linear Search in a given list of integers.
PROGRAM C CODE:
```
#include <stdio.h>
int main() {
   int array[100], search, c, n;
   printf("Enter number of elements in array\n");
   scanf("%d", &n);
   printf("Enter %d integer(s)\n", n);
   for (c = 0; c < n; c++)
      scanf("%d", &array[c]);
   printf("Enter a number to search\n");
   scanf("%d", &search);
   for (c = 0; c < n; c++)
   {
      if (array[c] == search)
      {
         printf("%d is present at location %d.\n", search, c+1);
         break;
      }
   }
   if (c == n)
   printf("%d isn't present in the array.\n", search);
   return 0;
}
```
OUTPUT:

```
Enter number of elements in array
8
Enter 8 integer(s)
4 88 64 8 19 37 0 2
Enter a number to search
0
0 is present at location 7.
```

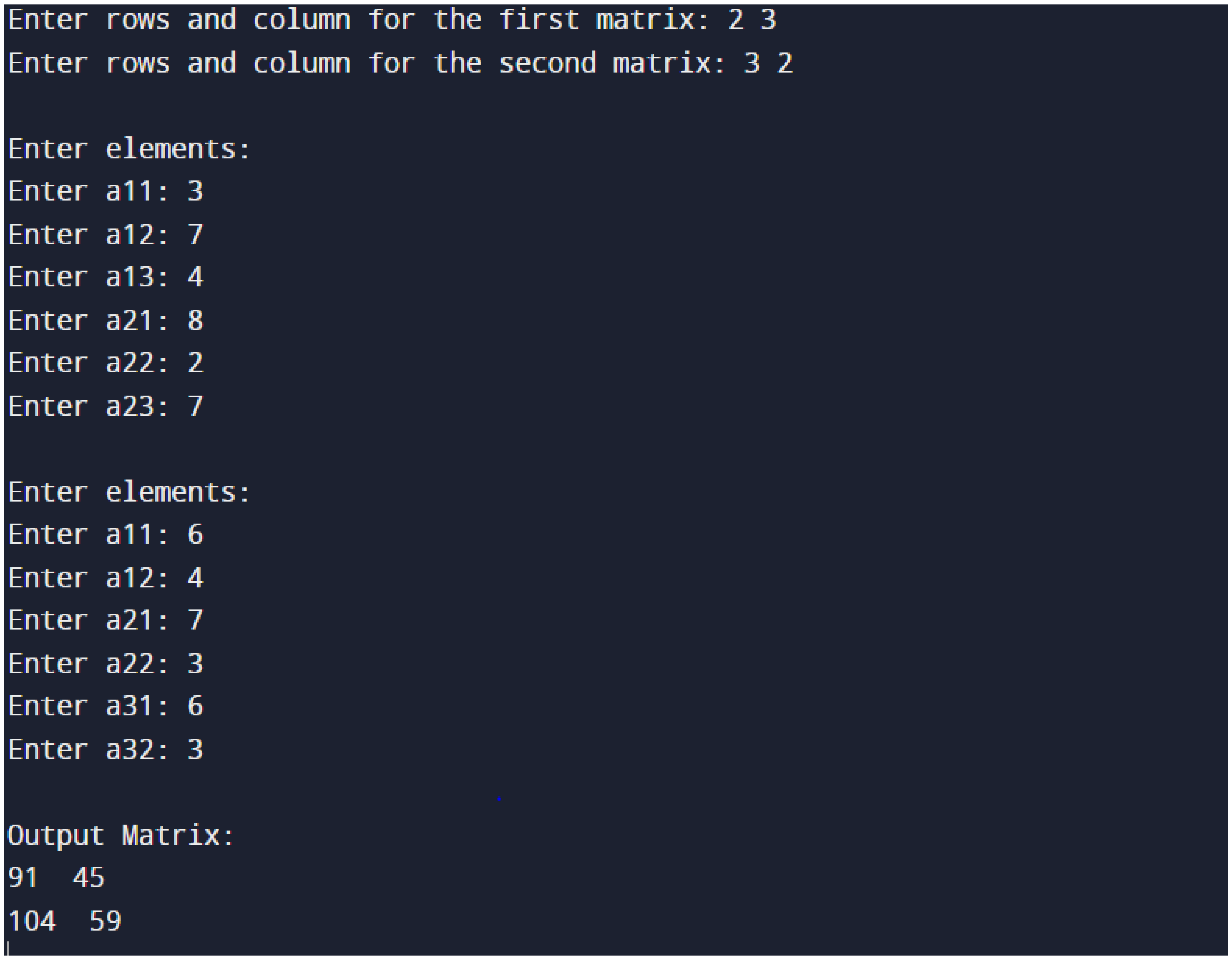
(**c**)
AIM: To find product of two matrices, and check for multicability between two matrices.
PROGRAM C CODE:

```
#include <stdio.h>
void getMatrixElements(int matrix[][10], int row, int column) {
   printf("\nEnter elements: \n");

   for (int i = 0; i < row; ++i)
   {
      for (int j = 0; j < column; ++j)
      {
         printf("Enter a%d%d: ", i + 1, j + 1);
         scanf("%d", &matrix[i][j]);
      }
  }
}
void multiplyMatrices(int first[][10],int second[][10],int result[][10],int r1,  int c1, int r2, int
c2) {
   for (int i = 0; i < r1; ++i)
   {
      for (int j = 0; j < c2; ++j)
      {
         result[i][j] = 0;
      }
   }
   for (int i = 0; i < r1; ++i) {
      for (int j = 0; j < c2; ++j)
      {
         for (int k = 0; k < c1; ++k)
         {
            result[i][j] += first[i][k] * second[k][j];
         }
      }
   }
}
void display(int result[][10], int row, int column) {
   printf("\nOutput Matrix:\n");
   for (int i = 0; i < row; ++i)
   {
      for (int j = 0; j < column; ++j)
```

```
        {
            printf("%d  ", result[i][j]);
            if (j == column - 1)
                printf("\n");
        }
    }
}
int main() {
    int first[10][10], second[10][10], result[10][10], r1, c1, r2, c2;
    printf("Enter rows and column for the first matrix: ");
    scanf("%d %d", &r1, &c1);
    printf("Enter rows and column for the second matrix: ");
    scanf("%d %d", &r2, &c2);
    while (c1 != r2)
    {
        printf("Error! Enter rows and columns again.\n");
        printf("Enter rows and columns for the first matrix: ");
        scanf("%d%d", &r1, &c1);
        printf("Enter rows and columns for the second matrix: ");
        scanf("%d%d", &r2, &c2);
   }
   getMatrixElements(first, r1, c1);
   getMatrixElements(second, r2, c2);
   multiplyMatrices(first, second, result, r1, c1, r2, c2);
   display(result, r1, c2);
   return 0;
}
```

OUTPUT:



```
Enter rows and column for the first matrix: 2 3
Enter rows and column for the second matrix: 3 2

Enter elements:
Enter a11: 3
Enter a12: 7
Enter a13: 4
Enter a21: 8
Enter a22: 2
Enter a23: 7

Enter elements:
Enter a11: 6
Enter a12: 4
Enter a21: 7
Enter a22: 3
Enter a31: 6
Enter a32: 3

Output Matrix:
91  45
104  59
```

SECTION 2:
THEORY:
The string can be defined as the one-dimensional array of characters terminated by a null ('\0'). The character array or the string is used to manipulate text such as words or sentences. Each character in the array occupies one byte of memory, and the last character must always be 0. The termination character ('\0') is important in a string since it is the only way to identify where the string ends. When we define a string as char s[10], the character s[10] is implicitly initialized with the null in the memory.
There are two ways to declare a string in c language.

- By char array
  Example: char ch[]={'a', 'b', 'c', 'd', '\0'};
- By string literal
  Example: char ch[]="abcd";

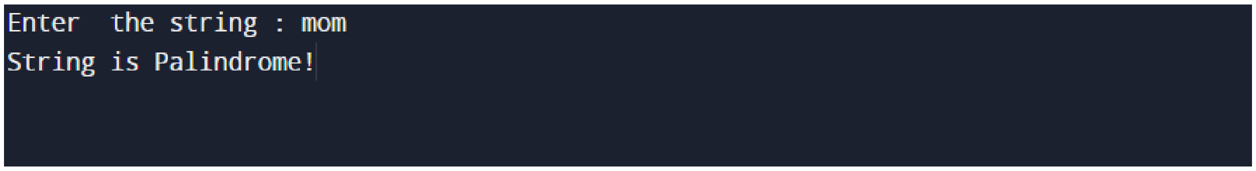
(**c**)

AIM: To check if the entered string is palindrome or not.

PROGRAM C CODE:

```
#include <stdio.h>
int main() {
    char s[1000];
    int i,n,c=0;
    printf("Enter  the string : ");
    gets(s);
    n=strlen(s);
    for(i=0;i<n/2;i++)
    {
        if(s[i]==s[n-i-1])
           c++;
           }
           if(c==i)
               printf("String is Palindrome!");
    else
        printf("String is not Palindrome!");
    return 0;
}
```

OUTPUT:



```
Enter  the string : mom
String is Palindrome!
```