DSA-ASSIGNMENT

Slot- L9+L10

Prof-Dr.Ramesh Ragala

Reg.no-22BCE1778

Name- Jayampu Akash Siva Teja.

LAB SHEET-1:

1.PAIR SUM:

CODE:

```
#include <stdio.h>
int main()
{
    int n,b;
    scanf("%d",&n);
    scanf("%d",&b);
    int a[n];
    for(int i=0;i<n;i++) {
        scanf("%d",&a[i]);
    }
    for(int i=0;i<n;i++) {
        for(int j=i+1;j<n-1;j++) {
            if(a[i]+a[j]==b) {
                printf("%d %d\n",a[i],a[j]);
            }
        }
    }
    return 0;
}
```

ALGORITHM:

```
Algorithm P_sum(A,n,b)
//A is an array of size n and target sum is b.
{
  for i←0 to n do {
     for j←i+1 to n-1 do {
         if (A[i]+A[j]==b) {
            write(A[i],A[j]);
          }
      }
  }
}
```

OUTPUT:

Output

```
/tmp/dU6d9upqBy.o
5
5
1 2 3 4 5
1 4
2 3
```

2.SYMETRIC MATRIX:

CODE:

```
#include <stdio.h>
int main() {
    int i, j;
    int n, m;
    scanf("%d %d", &n, &m);
    int a[n][m];
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            scanf("%d", &a[i][j]);
        }
    }
    int count = 0;
    for (i = 0; i < n; i++) {
        for (j = i+1; j < m; j++) {
            if (a[i][j] != a[j][i]) {
                count++;
            }
        }
    }
    if(count != 0 || n != m) {
        printf("Not Symmetric");
    }
    else{
        printf("Symmetric");
    }
```

```
    return 0;
}
```

ALGORITM:

```
Algorithm Symetric(n,m,a)  {
    for i ← 0 to n − 1 do
    {
        for j ← 0 to m − 1 do {
            read a[i][j]
        }
    }
    count ← 0
    for i ← 0 to n − 1 do
    {
        for j ← i + 1 to m − 1 do
        {
            if a[i][j] ≠ a[j][i] then {
                count ← count + 1
            }
        }
    }
   if count ≠ 0 or n ≠ m then {
        print "Not Symmetric"
    }
    else {
        print "Symmetric"
    }
}
```

OUTPUT:

```
/tmp/nAZyzDSXRx.o
3 3
1 2 3
4 5 6
7 8 9
Not Symmetric
```

3.REPEATED NUMBERS

CODE:

```
#include <stdio.h>
int main() {
    int n, i, j;
    scanf("%d", &n);
    int a[n];
    for(i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
    for(i = 0; i < n; i++) {
        int count = 0;
        for(j = 0; j < n; j++) {
            if(i != j && a[i] == a[j]) {
                count++;
            }
        }
        if(count == 0) {
            printf("%d ", a[i]);
        }
    }
    return 0;
}
```

ALGORITHM:

```
Algorithm remove(n, a)
//a is an array of size n
{
    for i ← 0 to n do
    {
        read a[i]
    }
    for i ← 0 to n do
     {
        count ← 0
        for j ← 0 to n do
        {
            if i ≠ j and a[i] = a[j] then
            {
                count ← count + 1
            }
        }
        if count = 0 then
            print a[i]
     }
```

OUTPUT:

```
Output
/tmp/nAZyzDSXRx.o
5
1 1 5 5 6
6
```

4.REMOVE DUPLICATE ELEMENTS:

CODE:

```
int main() {
    int n;
    scanf("%d", &n);
    int a[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
    int newSize = n;
   for (int i = 1; i < newSize; i++) {
        if (a[i] == a[i - 1]) {
            for (int j = i; j < newSize - 1; j++) {
                a[j] = a[j + 1];
            }
            newSize--;
            i--;
        }
    }

    printf("%d\n", newSize);
    for (int i = 0; i < newSize; i++) {
        printf("%d ", a[i]);
    }

    return 0;
}
```

ALGORITHM:

```
Repeated numbers(n, a)
//a is an array of size n.
{
    read n
    for i ← 0 to n − 1 do
    {
       read a[i]
    }
    newSize ← n

    for i ← 1 to newSize − 1 do
    {
        if a[i] = a[i − 1] then
        {
            for j ← i to newSize − 2 do
            {
                a[j] ← a[j + 1]
                newSize ← newSize − 1
                i ← i − 1
            }
            write(newsize)
   for i ← 0 to newSize − 1 do
     {
        write(a[i])
     }
}
```

OUTPUT:

Output

```
/tmp/dU6d9upqBy.o
5
2 2 2 2 2
1
2
```

5.SECOND SMALLEST NUMBER:

CODE:

```
#include <stdio.h>
int main(void)
{
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    int smallest = a[0],second_smallest=20000;
    for(int i=1;i<n;i++)
    {
       if(a[i]<smallest)
       {
        second_smallest=smallest;
        smallest =a[i];
       }
       else if(a[i] < second_smallest && a[i] !=
smallest)
        {
            second_smallest = a[i];
        }
    }
    printf("%d",second_smallest); }
```

ALGORITHM:

```
Algorithm Second_Smallest(n, a)
{
    read n
    for i ← 0 to n − 1 do
    {
        read a[i]
    }
    smallest ← a[0]
    second_smallest ← 20000
    for i ← 1 to n − 1 do
    {
        if a[i] < smallest then
        {
            second_smallest ← smallest
            smallest ← a[i]
        }
        else if a[i] < second_smallest and a[i] ≠
smallest then
        {
            second_smallest ← a[i]
        }
     write("Second smallest number:", second_smallest)
}
```

OUTPUT:

Output

```
/tmp/dU6d9upqBy.o
5
4 2 0 1 6
1
```

LAB SHEET-2:

1. Moving Negative numbers:

CODE:

```
#include <stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    int array[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&array[i]);
    }
    int
result[n],countpositive=0,countnegative=0,countzero=0;
    for (int i=0;i<n;i++)
    {
        if (array[i]<0)
        {
            countpositive++;
        }
        if(array[i]==0)
        {
           countzero++;
        }
    }
    countzero = countpositive-countzero-1;
```

```
    for(int i=0;i<n;i++)
    {
        if(array[i]<0)
        {
            result[countnegative]=array[i];
            countnegative++;
        }
        else if(array[i]>0)
        {
            result[countpositive]=array[i];
            countpositive++;
        }
        else if(array[i]==0)
        {
          result[countzero]=0;
          countzero++;
        }
    }
    for(int i=0; i<n;i++)
    {
        printf("%d ",result[i]);
    }
    return 0;
}
```

ALGORITHM:

```
Algorithm ReorganizeArray(n, array)
{
    read n

    for i ← 0 to n − 1 do
    {
        read array[i]
    }
    initialize result[n]
    countpositive ← 0
    countnegative ← 0
    countzero ← 0

    for i ← 0 to n − 1 do
    {
        if array[i] < 0 then
            countpositive ← countpositive + 1
        if array[i] = 0 then
            countzero ← countzero + 1
    }
    countzero ← countpositive − countzero − 1

    for i ← 0 to n − 1 do
    {
        if array[i] < 0 then
            result[countnegative] ← array[i]
```

```
            countnegative ← countnegative + 1
        else if array[i] > 0 then
            result[countpositive] ← array[i]
            countpositive ← countpositive + 1
        else if array[i] = 0 then
            result[countzero] ← 0
            countzero ← countzero + 1
    }
    for i ← 0 to n − 1 do
    {
       write(result[i])
    }
}
```

OUTPUT:

```
Output
/tmp/dU6d9upqBy.o
5
1 25 35 -12 7
-12 1 25 35 7
```

2. Rotations in Array:

CODE:

```
#include <stdio.h>
int main(void)
{
    int n,r;
    scanf("%d",&n);
    scanf("%d",&r);
    int array[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&array[i]);
    }
    while(r!=0)
    {
        int temp=array[0];
        for(int i=0;i<n-1;i++)
        {
            array[i]=array[i+1];
        }
        array[n-1]=temp;
        r--;
    }
    for(int i=0;i<n;i++)
    {
        printf("%d ",array[i]);
    }
```

```
}
```

ALGORITHM:

```
Algorithm RotateArrayRight(n, r, array)
{
    read n
    read r

    for i ← 0 to n − 1 do
    {
        read array[i]
    }
    while r ≠ 0 do
    {
        temp ← array[0]
        for i ← 0 to n − 2 do
            array[i] ← array[i + 1]
        array[n − 1] ← temp
        r ← r − 1
    }
    for i ← 0 to n − 1 do
    {
        Write(array[i])
    }
}
```

OUTPUT:

```
Output
/tmp/8DYJNPHXRK.o
5 2
1 2 3 4 5
3 4 5 1 2
```

2. Rotations in Array:

CODE:

```
#include <stdio.h>
int main(void)
{   int n,r;
    scanf("%d",&n);
    scanf("%d",&r);
    int array[n];
    for(int i=0;i<n;i++)
    {
        scanf("%d",&array[i]);
    }
    if(n%2==0)
    {
    while(r!=0)
    {
        int temp=array[0];
        for(int i=0;i<n-1;i++)
        {
            array[i]=array[i+1];
        }
```

```
        array[n-1]=temp;
        r--;
    }
    for(int i=0;i<n;i++)
    {
        printf("%d ",array[i]);
    }
    }
    else
    {
        while(r!=0)
    {
        int temp=array[n-1];
        for(int i=n-1;i>0;i--)
        {
            array[i]=array[i-1];
        }
        array[0]=temp;
        r--;
    }
    for(int i=0;i<n;i++)
    {
        printf("%d ",array[i]);
    }

    }
}
```

ALGORITHM:

```
RotateArrayRight(n, r, array) {
    read n
    read r

    for i ← 0 to n − 1 do {
        read array[i]
    }
    if n is even then {
        while r ≠ 0 do {
            temp ← array[0]
            for i ← 0 to n − 2 do
                array[i] ← array[i + 1]
            array[n − 1] ← temp
            r ← r − 1
         }
    }
    Else {
        while r ≠ 0 do {
            temp ← array[n − 1]
            for i ← n − 1 to 1 do
                array[i] ← array[i − 1]
            array[0] ← temp
            r ← r − 1
        }
    }
    for i ← 0 to n − 1 do {
```

```
        write(array[i])
    }
}
```

OUTPUT:

```
Output
/tmp/dU6d9upqBy.o
5
2
1 2 3 4 5
4 5 1 2 3
```

Staircase Case problem:

CODE:

```
#include <stdio.h>
int main(void)
{
    int n;
    scanf("%d",&n);
    int array[n],possible=1,currentstep=0;
    for(int i=0;i<n;i++)
    {
        scanf("%d",&array[i]);
    }
    while(possible==1 && currentstep<n-1)
    {
        currentstep+=array[currentstep];
```

```
        if((currentstep == n-1) || (((n-1)-
currentstep)<array[currentstep]) ||
array[currentstep]==0)
        {
            possible=0;
        }
    }
    if(currentstep==n-1)
    {
        printf("you have sucessfully reached the top
most step");
    }
    else
    {
        printf("you were not able to get to the top
most step\n");
        printf("you got stuck on step no
%d",currentstep+1);
    }
}
```

ALGORITHM:

```
Algorithm Steps(n, array)
{
    read n

    for i ← 0 to n − 1 do {
        read array[i]
    }
    possible ← 1
```

```
    currentstep ← 0


    while possible = 1 and currentstep < n − 1 do {
        currentstep ← currentstep + array[currentstep]
        if currentstep = n − 1 or (n − 1 − currentstep)
< array[currentstep] or array[currentstep] = 0 then
            possible ← 0
    }
    if currentstep = n − 1 then {
        write( "You have successfully reached the
topmost step.")
    }
    else {
        write( "You were not able to get to the topmost
step.")
        write ("You got stuck on step no", currentstep
+ 1)
    }
}
```

OUTPUT:

```
Output
/tmp/dU6d9upqBy.o
5
1 1 1 1 1
you have sucessfully reached the top most step
```

Next Greater Element

CODE:

```
#include <stdio.h>

int main(void)
{
    int n;
    scanf("%d", &n);
    int array[n], nextgreater[n];

    for(int i = 0; i < n; i++)
    {
        scanf("%d", &array[i]);
    }

    for(int i = 0; i < n; i++)
    {
        nextgreater[i] = -1;
        for(int j = i + 1; j < n; j++)
        {
            if(array[j] > array[i])
            {
                nextgreater[i] = array[j];
                break;
            }
        }
    }
```

```
    for(int i = 0; i < n; i++)
    {
        printf("%d ", nextgreater[i]);
    }

    return 0;
}
```

ALGORITHM:

```
Algorithm NextGreater(n, array) {
    read n

    create array[n]
    create nextgreater[n]

    for i ← 0 to n − 1 do {
        read array[i]
    }
    for i ← 0 to n − 1 do  {
        nextgreater[i] ← −1
        for j ← i + 1 to n − 1 do  {
            if array[j] > array[i] then {
                nextgreater[i] ← array[j]
                break
            }
        }
    }
```

```
    for i ← 0 to n − 1 do {
        write(nextgreater[i])
    }
}
```

OUTPUT:

```
Output
/tmp/dU6d9upqBy.o
5
1 2 3 4 5
2 3 4 5 -1
```

Revision on array

Code 1-

```
#include<stdio.h>
int main(void)
{
    int x[20],sum,i,n;
    printf("How many numbers");
    scanf("%d",&n);
    for (i=1;i<=n;i++)
    {
        scanf("%d",&x[i]);
    }
    sum=0;
    for (i=1;i<=n-2;i++)
    {
        sum=sum+(x[i]+x[i+1])*x[i+2];
    }
    printf("%d",sum);
}
```

Code 2-

```
#include<stdio.h>
int main(void)
{
    int x[20],ans,i,n;
    printf("How many numbers");
    scanf("%d",&n);
```

```
    for (i=1;i<=n;i++)
    {
        scanf("%d",&x[i]);
    }
    ans=1;
    for (i=1;i<=n-2;i++)
    {
        ans*=(x[i]+x[i+2]);
    }
    printf("%d",ans);
}
```

Code 3-

```
#include<stdio.h>
int main(void)
{
    int x[20],ans,i,n;
    printf("How many numbers");
    scanf("%d",&n);
    for (i=1;i<=n;i++)
    {
        scanf("%d",&x[i]);
    }
    ans=0;
    for (i=1;i<=n-2;i++)
    {
        ans+=(x[i]-x[i+1])*(x[i+1]+x[i+2]);
    }
    printf("%d",ans);
}
```

Code4:-

```
#include<stdio.h>
int main(void)
{
    int x[20],ans,i,n;
    printf("How many numbers");
    scanf("%d",&n);
    for (i=0;i<n;i++)
```

```
    {
        scanf("%d",&x[i]);
    }
    int rev=n-1;
    ans=0;
    for (i=0;i<n;i++)
    {
        ans+=x[i]*x[rev];
        rev--;
    }
    printf("%d",ans);
}
```

CODE-5:

```
#include<stdio.h>
int main(void)
{
    int x[20],ans,i,n,mul=1;
    printf("How many numbers");
    scanf("%d",&n);
    for (i=0;i<n;i++)
    {
        scanf("%d",&x[i]);
    }
    ans=0;
    for (i=0;i<n-1;i++)
```

```
    {
       ans+=mul*(x[i]+x[i+1]);
       mul+=1;
    }
    printf("%d",ans);
}
```

CODE-6:

```
#include<stdio.h>
#include<math.h>
int main(void)
{
    int x[20],ans,i,n,po=1;
    printf("How many numbers");
    scanf("%d",&n);
    for (i=0;i<n;i++)
    {
        scanf("%d",&x[i]);
    }
    ans=0;
    for (i=0;i<n-1;i++)
    {
        ans+=pow((x[i]+x[i+1]),po);
        po++;
    }
    printf("%d",ans);
}
```

CODE-7:

```
#include<stdio.h>
int main(void)
{
    int n,m;
    scanf("%d",&m);
    scanf("%d",&n);
    int matrix[m][n];
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        {
            scanf("%d",&matrix[i][j]);
        }
    }
    for(int i=0;i<m;i++)
    {
        if(i<n)
        {
            printf("%d",matrix[i][i]);
        }
    }

}
```

CODE-8:

```
#include<stdio.h>
int main(void)
{
    int n,m;
    scanf("%d",&m);
    scanf("%d",&n);
    int matrix[m][n],matrix2[m][n],resultmatrix[m][n];
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        {
            scanf("%d",&matrix[i][j]);
        }
    }
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        {
            scanf("%d",&matrix2[i][j]);
        }
    }
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        {

resultmatrix[i][j]=matrix[i][j]+matrix2[i][j];
```

```
        }
    }
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        {
            printf("%d ",resultmatrix[i][j]);
        }
        printf("\n");
    }



}
```

CODE-9:

```
#include<stdio.h>
int main(void)
{
    int r1,c1,r2,c2;
    scanf("%d",&r1);
    scanf("%d",&c1);
    scanf("%d",&r2);
    scanf("%d",&c2);
    int
matrix[r1][c1],matrix2[r2][c2],resultmatrix[r1][c2];
    for(int i=0;i<r1;i++)
    {
        for(int j=0;j<c1;j++)
```

```
        {
            scanf("%d",&matrix[i][j]);
        }
    }
    for(int i=0;i<r2;i++)
    {
        for(int j=0;j<c2;j++)
        {
            scanf("%d",&matrix2[i][j]);
        }
    }
    for (int i = 0; i < r1; ++i) {
      for (int j = 0; j < c2; ++j) {
        resultmatrix[i][j]=0;
         for (int k = 0; k < c1; ++k) {
            resultmatrix[i][j] += matrix[i][k] *
matrix2[k][j];
         }
      }
   }
   for(int i=0;i<r1;i++)
   {
    for(int j=0;j<c2;j++)
    {
        printf("%d ",resultmatrix[i][j]);
    }
    printf("\n");
```

```
    }
}
```

CODE-10:

```
#include<stdio.h>
int main(void)
{
    int n,m;
    scanf("%d",&m);
    scanf("%d",&n);
    int matrix[m][n];
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        {
            scanf("%d",&matrix[i][j]);
        }
    }
    for(int i=0;i<m;i++)
    {
        int min=matrix[i][0];
        for(int j=1;j<n;j++)
        {
            if(matrix[i][j]<min)
            {
                min= matrix[i][j];
            }
```

```
        }
        printf("the least value of row no%d is
%d\n",i+1,min);


    }
}
```

CODE-11:

```
#include <stdio.h>

int main() {
    int rows, cols;

    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    printf("Enter the number of columns: ");
    scanf("%d", &cols);

    int matrix[rows][cols];

    printf("Enter the matrix elements:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
```

```
    int row_min[rows];
    int max_of_min = matrix[0][0];

    for (int i = 0; i < rows; i++) {
        int min = matrix[i][0];
        for (int j = 1; j < cols; j++) {
            if (matrix[i][j] < min) {
                min = matrix[i][j];
            }
        }
        row_min[i] = min;
        if (min > max_of_min) {
            max_of_min = min;
        }
    }

    printf("Minimum elements in each row:\n");
    for (int i = 0; i < rows; i++) {
        printf("Row %d: %d\n", i + 1, row_min[i]);
    }

    printf("Maximum of all minimum elements: %d\n",
max_of_min);

    return 0;
}
```

CODE-12:

```c
#include <stdio.h>

int main() {
    int rows, cols;

    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    printf("Enter the number of columns: ");
    scanf("%d", &cols);

    double matrix[rows][cols];

    printf("Enter the matrix elements:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%lf", &matrix[i][j]);
        }
    }

    for (int i = 0; i < rows; i++) {
        if (matrix[i][i] != 0) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] /= matrix[i][i];
            }
        }
```

```
    }

    printf("Matrix after division by diagonal
elements:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%.2lf ", matrix[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

CODE-13:

```
#include <stdio.h>

void printMatrix(float **matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%.2f ", matrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int rows, cols;
```

```
    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    printf("Enter the number of columns: ");
    scanf("%d", &cols);

    float **matrix = (float **)malloc(rows *
sizeof(float *));
    for (int i = 0; i < rows; i++) {
        matrix[i] = (float *)malloc(cols *
sizeof(float));
    }

    printf("Enter the matrix elements:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%f", &matrix[i][j]);
        }
    }

    int x, y;
    float p;

    printf("Enter the values of x, y, and p: ");
    scanf("%d %d %f", &x, &y, &p);
```

```
    if (x >= 0 && x < rows && y >= 0 && y < rows) {
        for (int j = 0; j < cols; j++) {
            matrix[x][j] -= p * matrix[y][j];
        }

        printf("Matrix after row operation:\n");
        printMatrix(matrix, rows, cols);
    } else {
        printf("Invalid row indices.\n");
    }

    for (int i = 0; i < rows; i++) {
        free(matrix[i]);
    }
    free(matrix);

    return 0;
}
```

CODE-14:

```
#include <stdio.h>

void rowOperation(double matrix[2][2], int row, double
p) {
    for (int j = 0; j < 2; j++) {
        matrix[row][j] -= p * matrix[1][j];
```

```
    }
}

int main() {
    double matrix[2][2];
    int x, y;

    printf("Enter the value of x: ");
    scanf("%d", &x);

    printf("Enter the value of y: ");
    scanf("%d", &y);

    printf("Enter the elements of the 2x2 matrix:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            scanf("%lf", &matrix[i][j]);
        }
    }

    double p = matrix[0][0] != 0 ? -matrix[1][0] /
matrix[0][0] : 0;

    rowOperation(matrix, 1, p * x * y);

    printf("Matrix after row operation:\n");
    for (int i = 0; i < 2; i++) {
```

```
        for (int j = 0; j < 2; j++) {
            printf("%lf ", matrix[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

CODE-15:

```
#include <stdio.h>

void rowOperation(double matrix[2][2], int row, double
p) {
    for (int j = 0; j < 2; j++) {
        matrix[row][j] -= p * matrix[row - 1][j];
    }
}

int main() {
    int nr;
    double matrix[2][2];

    printf("Enter the number of rows (nr): ");
    scanf("%d", &nr);

    if (nr < 2) {
```

```
        printf("Number of rows should be at least
2.\n");
        return 1;
    }

    printf("Enter the elements of the 2x2 matrix:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            scanf("%lf", &matrix[i][j]);
        }
    }

    for (int y = 1; y <= nr; y++) {
        for (int x = 1; x <= nr; x++) {
            if (x != y) {
                double p = matrix[0][0] != 0 ? -
matrix[x - 1][0] / matrix[0][0] : 0;
                rowOperation(matrix, x, p * y);
            }
        }
    }

    printf("Matrix after row operations:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            printf("%lf ", matrix[i][j]);
        }
```

```
        printf("\n");
    }

    return 0;
}
```

CODE-16:

```
#include <stdio.h>

void rowOperation(double matrix[2][2], int row, double
p) {
    for (int j = 0; j < 2; j++) {
        matrix[row][j] -= p * matrix[row - 1][j];
    }
}

int main() {
    int nr;
    double matrix[2][2];

    printf("Enter the number of rows (nr): ");
    scanf("%d", &nr);

    if (nr < 2) {
        printf("Number of rows should be at least
2.\n");
        return 1;
    }
```

```
    printf("Enter the elements of the 2x2 matrix:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            scanf("%lf", &matrix[i][j]);
        }
    }

    for (int y = 1; y <= nr; y++) {
        for (int x = 1; x <= nr; x++) {
            if (x != y) {
                double p = matrix[0][0] != 0 ? -
matrix[x - 1][0] / matrix[0][0] : 0;
                rowOperation(matrix, x, p * y);
            }
        }
    }

    printf("Matrix after row operations:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            printf("%lf ", matrix[i][j]);
        }
        printf("\n");
    }

    return 0;
```

```
}
```

CODE-17:

```
#include <stdio.h>

void printMatrix(double matrix[][10], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= n; j++) {
            printf("%.2lf ", matrix[i][j]);
        }
        printf("\n");
    }
}

void solveEquations(double matrix[][10], int n) {
    for (int i = 0; i < n; i++) {
        if (matrix[i][i] == 0.0) {
            printf("Divide by zero encountered!\n");
            return;
        }

        for (int j = i + 1; j < n; j++) {
            double ratio = matrix[j][i] / matrix[i][i];
            for (int k = 0; k <= n; k++) {
                matrix[j][k] -= ratio * matrix[i][k];
            }
        }
```

```
    }

    for (int i = n - 1; i >= 0; i--) {
        matrix[i][n] /= matrix[i][i];
        matrix[i][i] = 1.0;

        for (int j = i - 1; j >= 0; j--) {
            matrix[j][n] -= matrix[j][i] *
matrix[i][n];
            matrix[j][i] = 0.0;
        }
    }

    printf("Solution:\n");
    for (int i = 0; i < n; i++) {
        printf("x%d = %.2lf\n", i + 1, matrix[i][n]);
    }
}

int main() {
    int nr, nc;
    double matrix[10][10];

    printf("Enter the number of rows (nr): ");
    scanf("%d", &nr);

    printf("Enter the number of columns (nc): ");
```

```
    scanf("%d", &nc);

    if (nc != nr + 1) {
        printf("Number of columns should be equal to
number of rows + 1.\n");
        return 1;
    }

    printf("Enter the augmented matrix:\n");
    for (int i = 0; i < nr; i++) {
        for (int j = 0; j <= nc; j++) {
            scanf("%lf", &matrix[i][j]);
        }
    }

    solveEquations(matrix, nr);

    return 0;
}
```

CODE-18:

```
#include<stdio.h>

int main(){

  int a[3][3],i,j;
  float determinant=0;
```

```
  printf("Enter the 9 elements of matrix: ");
  for(i=0;i<3;i++)
      for(j=0;j<3;j++)
           scanf("%d",&a[i][j]);

  printf("\nThe matrix is\n");
  for(i=0;i<3;i++){
      printf("\n");
      for(j=0;j<3;j++)
           printf("%d\t",a[i][j]);
  }

  for(i=0;i<3;i++)
      determinant = determinant +
(a[0][i]*(a[1][(i+1)%3]*a[2][(i+2)%3] -
a[1][(i+2)%3]*a[2][(i+1)%3]));

   printf("\nInverse of matrix is: \n\n");
   for(i=0;i<3;i++){
      for(j=0;j<3;j++)
           printf("%.2f\t",((a[(i+1)%3][(j+1)%3] *
a[(i+2)%3][(j+2)%3]) -
(a[(i+1)%3][(j+2)%3]*a[(i+2)%3][(j+1)%3]))/
determinant);
       printf("\n");
   }
 return 0;
}
```