

Subject Name: Computer Programming with C

Subject Code: MCA102

Assignment-3

Topic: Arrays and Structures

Name: Shashwat Khaitan

Section: B

Enrollment No: 12024006015093

Class Roll No: 36

1. Write a program to store marks for n numbers of students in an array and print their marks.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n,i;
```

```
    printf("enter the number of students:");
```

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

```
    int marks[n];
```

```
    printf("enter the marks of %d student^\n", n);
```

```
    for(i=0;i<n;i++)
```

```
{  
    scanf("%d", &marks[i]);  
}  
  
printf("marks of students are:\n");  
for(i=0;i<n;i++)  
{  
    printf("%d", marks[i]);  
}  
  
return 0;  
}
```

OUTPUT

```
root@SK-PC: ~  
root@SK-PC:~# vim marks.c  
root@SK-PC:~# gcc -o s marks.c  
root@SK-PC:~# ./s  
enter the number of students:  
2  
enter the marks of 2 students:  
90  
100  
marks of students are:  
90100root@SK-PC:~#
```

2. Write a program that stores the marks of the subjects Mathematics and English of n number of students in an array and then prints their total marks.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n,i;
```

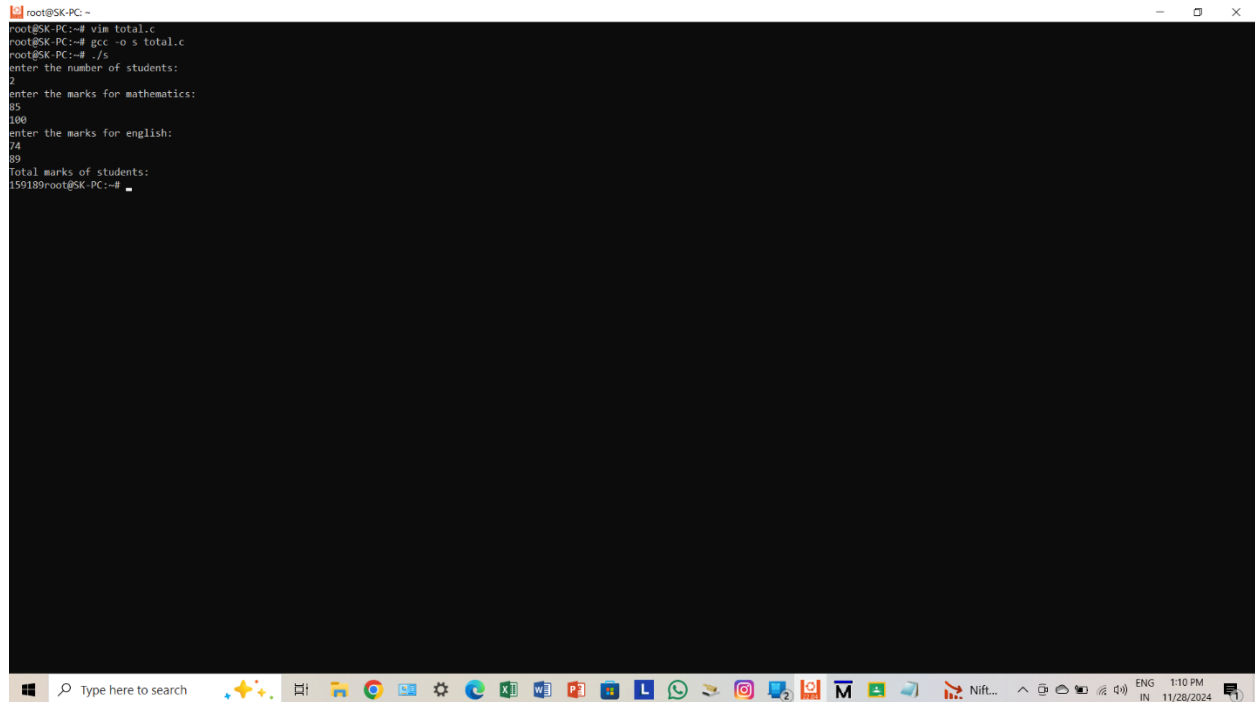
```
    printf("enter the number of students: ");
```

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

```
    int m[n], e[n], total[n];
```

```
printf("enter the marks for mathematics:\n");  
  
for(i=0;i<n;i++)  
{  
    scanf("%d", &m[i]);  
}  
  
printf("enter the marks for english:\n");  
  
for(i=0;i<n;i++)  
{  
    scanf("%d", &e[i]);  
}  
  
printf("Total marks of students:\n");  
  
for(i=0;i<n;i++)  
{  
    total[i]= m[i]+e[i];  
    printf("%d", total[i]);  
}  
  
return 0;  
}
```

OUTPUT



```
root@SK-PC: ~  
root@SK-PC:~# vim total.c  
root@SK-PC:~# gcc -o s total.c  
root@SK-PC:~# ./s  
enter the number of students:  
2  
enter the marks for mathematics:  
85  
100  
enter the marks for english:  
74  
89  
Total marks of students:  
159189root@SK-PC:~#
```

3. Write a program to insert an element in an array in a particular position.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n,i,pos,v;
```

```
    printf("enter the size of array: ");
```

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

```
    int arr[n+1];
```

```
printf("enter %d elements:\n", n);

for(i=0;i<n;i++)

{

    scanf("%d", &arr[i]);

}

printf("enter the position and value to insert: ");

scanf("%d %d", &pos, &v);

for(i=n;i>=pos;i--)

{

    arr[i]=arr[i-1];

}

arr[pos-1]=v;

printf("array after insertion:\n");

for(i=0;i<=n;i++)

{

    printf("%d", arr[i]);

}

return 0;

}
```

OUTPUT

```
root@SK-PC: ~  
root@SK-PC:~# vi insert.c  
root@SK-PC:~# gcc -o s insert.c  
root@SK-PC:~# ./s  
enter the size of array:  
5  
enter 5 elements:  
10  
20  
30  
40  
50  
enter the position and value to insert:  
2  
15  
array after insertion:  
101520304050root@SK-PC:~#
```

4. Write a program to delete an element from a particular position of an array.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n,i,pos;
```

```
    printf("enter the size of array: ");
```

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

```
    int a[n];
```

```
printf("enter %d elements:\n", n);

for(i=0;i<n;i++)

{

    scanf("%d", &a[i]);

}


printf("enter the position to delete: ");

scanf("%d", &pos);

for(i=pos-1;i<n-1;i++)

{

    a[i]=a[i+1];

}


printf("Array after deletion:\n");

for(i=0;i<n-1;i++)

{

    printf("%d ", a[i]);

}


return 0;

}
```


OUTPUT

```
root@SK-PC: ~  
root@SK-PC:~# vim delete.c  
root@SK-PC:~# gcc -o s delete.c  
root@SK-PC:~# ./s  
enter the size of array:  
5  
enter 5 elements:  
85  
96  
100  
78  
52  
enter the position to delete:  
4  
Array after deletion:  
85 96 100 52 root@SK-PC:~#
```

5. Write a program to convert a decimal number taken as input from a user to the corresponding binary number and store the result in an array.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n, b[32], i=0;
```

```
    printf("enter a decimal number: ");
```

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

```
    while(n>0)
```

```
{  
    b[i]=n%2;  
    n=n/2;  
    i++;  
}  
printf("Binary equivalent: ");  
for(i=i-1;i>=0;i--)  
{  
    printf("%d", b[i]);  
}  
  
return 0;  
}
```

OUTPUT

```
root@SK-PC: ~  
root@SK-PC:~# vim converse.c  
root@SK-PC:~# gcc -o 5 converse.c  
root@SK-PC:~# ./5  
enter a decimal number: 20  
Binary equivalent: 10100root@SK-PC:~#
```

6. Write a program to input a binary number in an array and convert it into a corresponding decimal number.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n,d=0,b=1,digit,i;
```

```
    printf("enter the number of binary digits: ");
```

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

```
    int binary[n];
```

```
    printf("enter the binary number:\n");
```

```
for(i=0;i<n;i++)  
{  
    scanf("%d", &binary[i]);  
}  
for(i=n-1;i>=0;i--)  
{  
    digit=binary[i];  
    d=d+digit*b;  
    b=b*2;  
}  
  
printf("Decimal equivalent: %d\n", d);  
return 0;  
}
```

OUTPUT

```
root@SK-PC: ~  
root@SK-PC:~# vim conversion1.c  
root@SK-PC:~# gcc -o s conversion1.c  
root@SK-PC:~# ./s  
enter the number of binary digits: 2  
enter the binary number:  
11  
10  
Decimal equivalent: 32  
root@SK-PC:~#
```

7. Write a program to find the smallest and the largest elements in an array.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n,i,large,small;
```

```
    printf("enter the size of array: ");
```

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

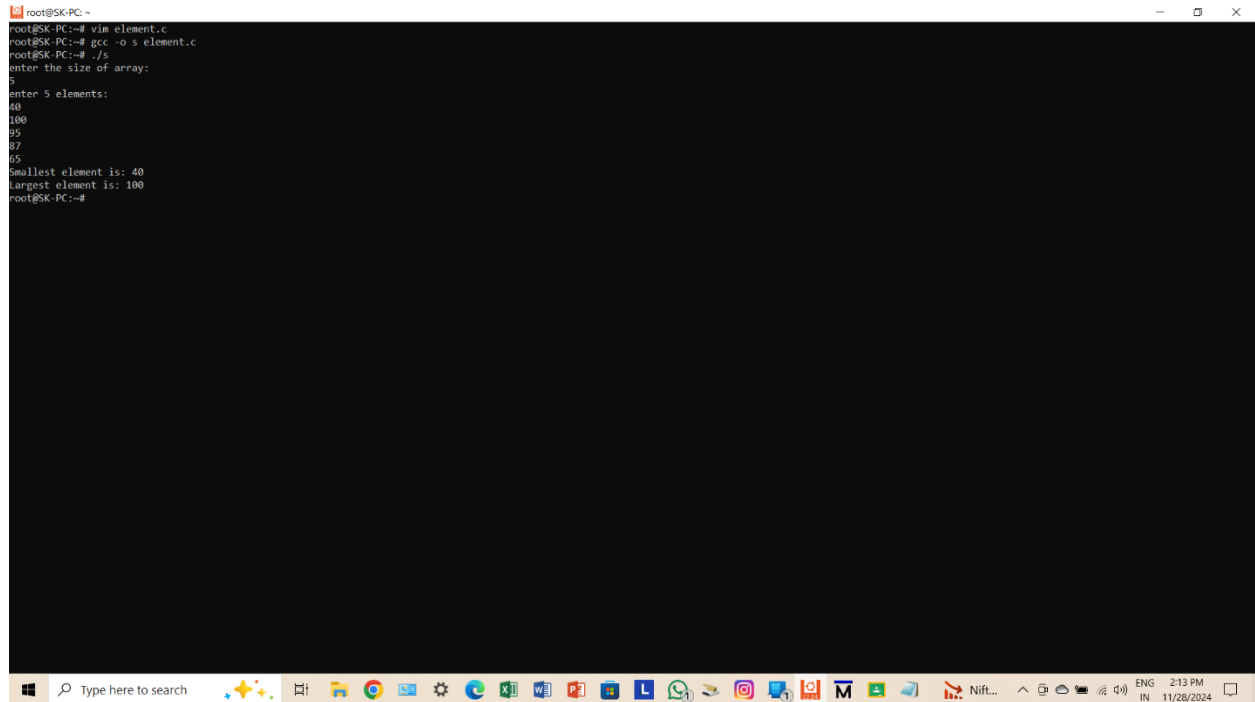
```
    int arr[n];
```

```
    printf("enter %d elements:\n", n);
```

```
    for(i=0;i<n;i++)
```

```
{  
    scanf("%d", &arr[i]);  
}  
small=large=arr[0];  
for(i=1;i<n;i++)  
{  
    if(arr[i]<small)  
    {  
        small=arr[i];  
    }  
    if(arr[i]>large)  
    {  
        large=arr[i];  
    }  
}  
printf("Smallest element is: %d\n", small);  
printf("Largest element is: %d\n", large);  
  
return 0;  
}
```

OUTPUT



```
root@SK-PC: ~  
root@SK-PC:~# vim element.c  
root@SK-PC:~# gcc -o s element.c  
root@SK-PC:~# ./s  
enter the size of array:  
5  
enter 5 elements:  
40  
100  
95  
87  
65  
Smallest element is: 40  
Largest element is: 100  
root@SK-PC:~#
```

8. Write a program for deleting duplicate elements in an array.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n,i,j,k;
```

```
    printf("enter the size of array: ");
```

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

```
    int arr[n];
```

```
printf("enter %d elements:\n", n);
```

```
for(i=0;i<n;i++)
```

```
{
```

```
    scanf("%d", &arr[i]);
```

```
}
```

```
for(i=0;i<n;i++)
```

```
{
```

```
    for(j=i+1;j<n;j)
```

```
    {
```

```
        if(arr[i]==arr[j])
```

```
        {
```

```
            for(k=j;k<n-i;k++)
```

```
            {
```

```
                arr[k]=arr[k+1];
```

```
            }
```

```
            n--;
```

```
        }
```

```
    else
```

```
    {
```

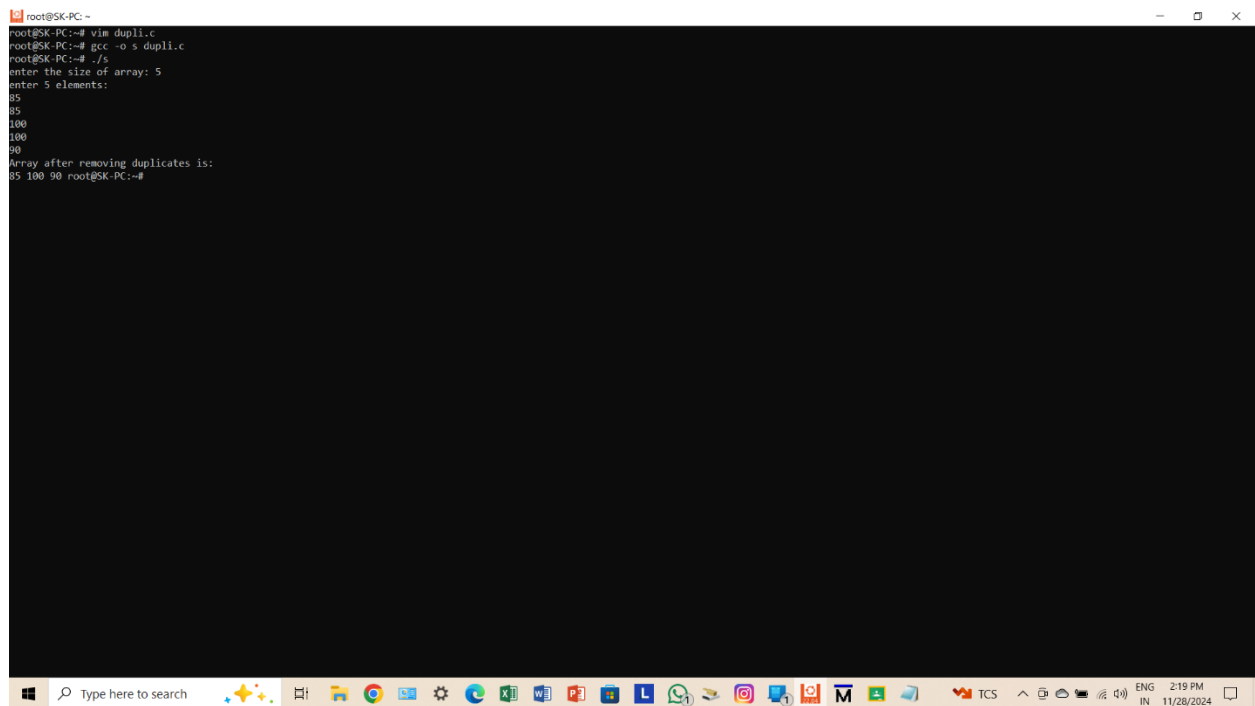
```
        j++;
```

```
    }
```



```
    }  
  
}  
  
printf("Array after removing duplicates is: \n");  
  
for(i=0;i<n;i++)  
  
{  
  
    printf("%d ", arr[i]);  
  
}  
  
return 0;  
  
}
```

OUTPUT



```
root@SK-PC:~  
root@SK-PC:~# vim dupli.c  
root@SK-PC:~# gcc -o s dupli.c  
root@SK-PC:~# ./s  
enter the size of array: 5  
enter 5 elements:  
85  
85  
100  
100  
90  
Array after removing duplicates is:  
85 100 90 root@SK-PC:~#
```

The screenshot shows a terminal window with a black background and white text. The user is in a root shell on a machine named SK-PC. They use the vim editor to create a file named dupli.c, then compile it with gcc into an executable named s. They run the program, which prompts for the size of the array (5) and then for 5 elements. The user enters 85, 85, 100, 100, and 90. The program outputs the array after removing duplicates, which is 85 100 90. The Windows taskbar is visible at the bottom of the screen.

9. Write a program to search for a particular element in an array.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n,i,s,f=0;
```

```
    printf("enter the size of array: ");
```

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

```
    int arr[n];
```

```
    printf("enter %d element:\n", n);
```

```
    for(i=0;i<n;i++)
```

```
    {
```

```
        scanf("%d", &arr[i]);
```

```
    }
```

```
    printf("enter the element to search: ");
```

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

```
    for(i=0;i<n;i++)
```

```
    {
```

```
        if(arr[i]==s)
```

```
        {
```

```
        printf("Element found at position: %d\n", i+1);

        f=1;

        break;

    }

}

if(!f)

{

    printf("Element not found.\n");

}


return 0;

}
```

OUTPUT

```
root@SK-PC:~# vim find.c
root@SK-PC:~# gcc -o s find.c
root@SK-PC:~# ./s
enter the size of array: 5
enter 5 element:
85
41
50
98
64
enter the element to search: 50
Element found at position: 3
root@SK-PC:~# gcc -o s find.c
root@SK-PC:~# ./s
enter the size of array: 4
enter 4 element:
85
74
99
100
enter the element to search: 5
Element not found.
root@SK-PC:~#
```

10. Write a program to sort n elements (in ascending order).

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n,i,j,t;
```

```
    printf("enter the size of array: ");
```

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

```
    int arr[n];
```

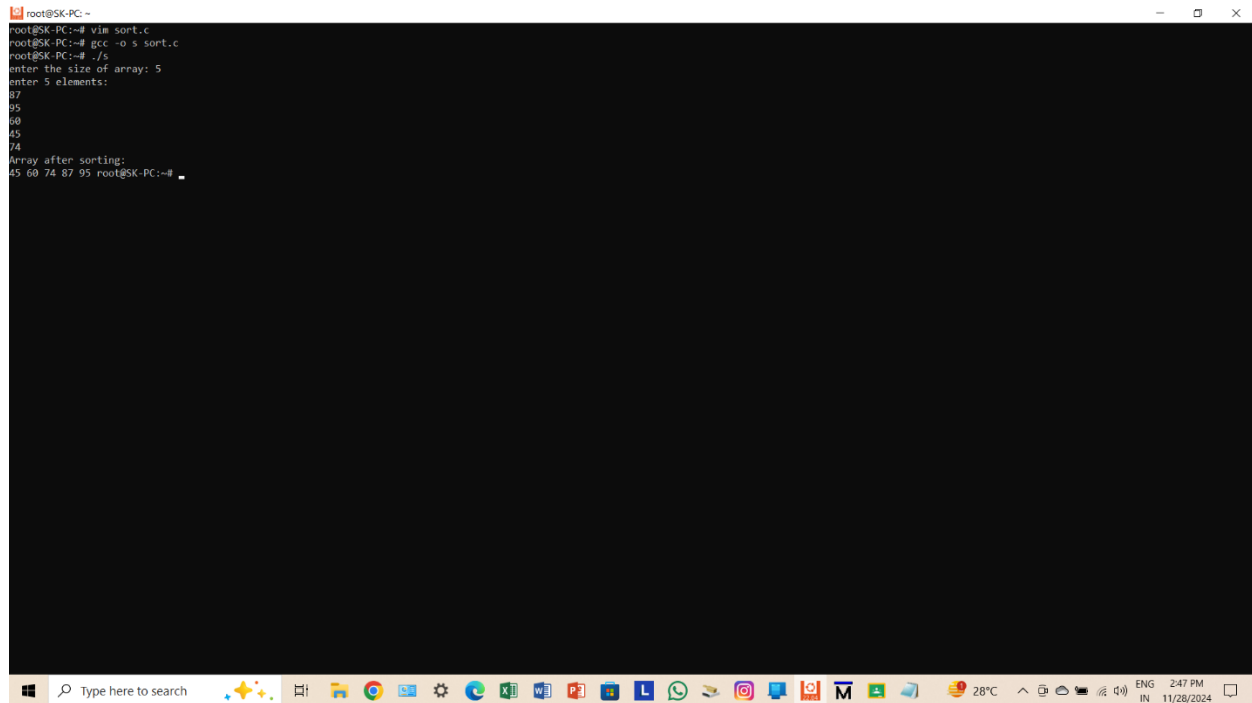
```
    printf("enter %d elements:\n", n);
```

```
    for(i=0;i<n;i++)
```

```
{  
    scanf("%d", &arr[i]);  
}  
for(i=0;i<n-1;i++)  
{  
    for(j=0;j<n-i-1;j++)  
    {  
        if(arr[j]>arr[j+1])  
        {  
            t=arr[j];  
            arr[j]=arr[j+1];  
            arr[j+1]=t;  
        }  
    }  
}  
printf("Array after sorting:\n");  
for(i=0;i<n;i++)  
{  
    printf("%d ", arr[i]);  
}
```

```
    return 0;  
}
```

OUTPUT



```
root@SK-PC: ~  
root@SK-PC:~# vim sort.c  
root@SK-PC:~# gcc -o s sort.c  
root@SK-PC:~# ./s  
enter the size of array: 5  
enter 5 elements:  
87  
95  
60  
45  
74  
Array after sorting:  
45 60 74 87 95 root@SK-PC:~#
```

11. Write a program to find the second-highest number from the array without using sorting.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n,i,large,second;
```

```
    printf("enter the size of array: ");
```

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

int arr[n];

printf("enter %d elements:\n", n);

for(i=0;i<n;i++)

{

    scanf("%d", &arr[i]);

}

large=second= -2147483648;

for(i=0;i<n;i++)

{

    if(arr[i]>large)

    {

        second=large;

        large=arr[i];

    }

    else if(arr[i]>second && arr[i]!=large)

    {

        second=arr[i];

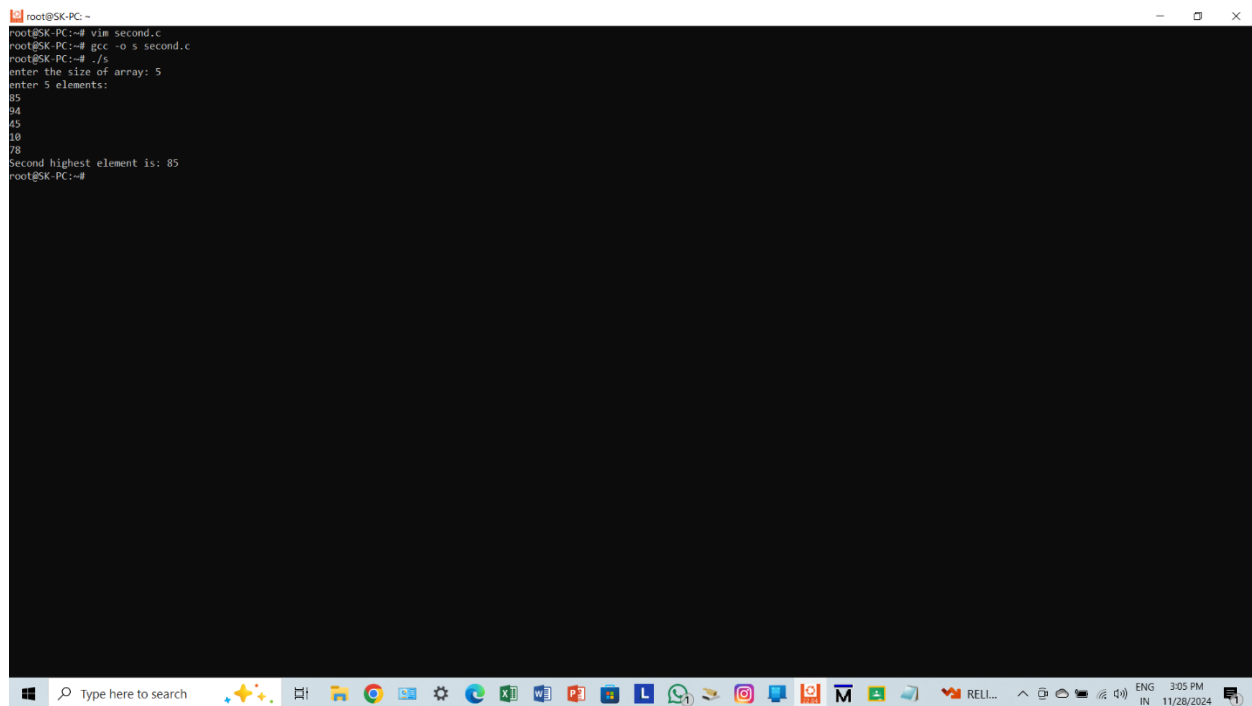
    }

}

if(second== -2147483648)
```

```
{  
  
    printf("No second highest element.\n");  
  
}  
  
else  
  
{  
  
    printf("Second highest element is: %d\n", second);  
  
}  
  
return 0;  
  
}
```

OUTPUT



```
root@SK-PC:~#  
root@SK-PC:~# vim second.c  
root@SK-PC:~# gcc -o s second.c  
root@SK-PC:~# ./s  
enter the size of array: 5  
enter 5 elements:  
85  
94  
45  
10  
78  
Second highest element is: 85  
root@SK-PC:~#
```


12. Write a program to perform addition and subtraction between two matrices.

```
#include <stdio.h>

int main()
{
    int r,c,i,j;

    printf("enter the number of rows and columns: ");
    scanf("%d %d", &r, &c);

    int m1[r][c], m2[r][c], sum[r][c], diff[r][c];

    printf("enter the elements of first matrix:\n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            scanf("%d", &m1[i][j]);
        }
    }

    printf("enter the elements of second matrix:\n");
    for(i=0;i<r;i++)
```

```
{  
    for(j=0;j<c;j++)  
    {  
        scanf("%d", &m2[i][j]);  
    }  
}  
printf("Sum of matrices is:\n");  
for(i=0;i<r;i++)  
{  
    for(j=0;j<c;j++)  
    {  
        sum[i][j]= m1[i][j]+m2[i][j];  
        printf("%d ", sum[i][j]);  
    }  
    printf("\n");  
}  
printf("Difference of matrices is:\n");  
for(i=0;i<r;i++)  
{  
    for(j=0;j<c;j++)  
    {
```

```
        diff[i][j]= m1[i][j]-m2[i][j];

        printf("%d ", diff[i][j]);

    }

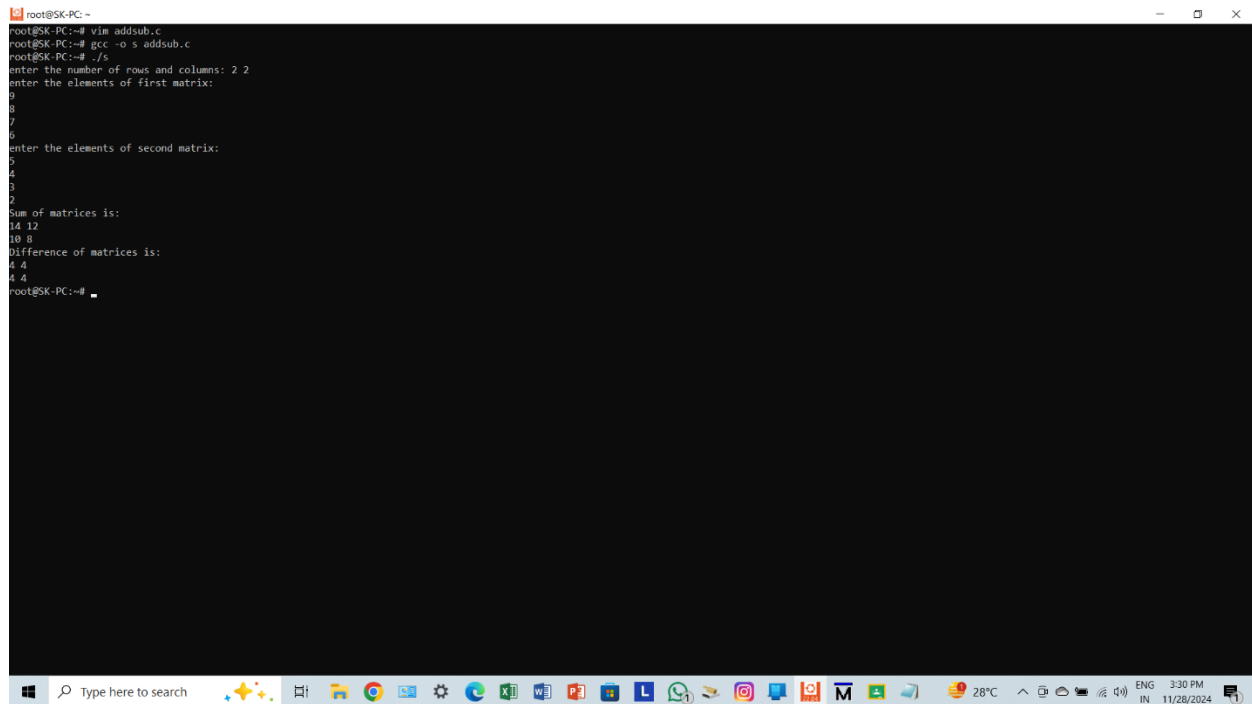
    printf("\n");

}

return 0;

}
```

OUTPUT



```
root@SK-PC: ~  
root@SK-PC:~# vim addsub.c  
root@SK-PC:~# gcc -o s addsub.c  
root@SK-PC:~# ./s  
enter the number of rows and columns: 2 2  
enter the elements of first matrix:  
9  
8  
7  
6  
enter the elements of second matrix:  
5  
4  
3  
2  
Sum of matrices is:  
14 12  
10 8  
Difference of matrices is:  
4 4  
4 4  
root@SK-PC:~#
```

The screenshot shows a terminal window with a black background and white text. The user is in a directory named '~' on a system named 'SK-PC'. They have edited a file 'addsub.c' using 'vim', compiled it with 'gcc -o s addsub.c', and executed it with './s'. The program prompts for the number of rows and columns (2 2), then for the elements of the first matrix (9, 8, 7, 6), and then for the elements of the second matrix (5, 4, 3, 2). It then displays the sum of the matrices (14 12, 10 8) and the difference of the matrices (4 4, 4 4). The terminal window has standard Linux window controls at the top right. The bottom of the image shows a Windows taskbar with various icons and system information like '28°C' and '3:30 PM'.

13. Write a program to transpose a matrix.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int r,c,i,j;
```

```
    printf("enter the number of rows and columns: ");
```

```
    scanf("%d %d", &r, &c);
```

```
    int m[r][c], t[c][r];
```

```
    printf("enter the elements of the matrix:\n");
```

```
    for(i=0;i<r;i++)
```

```
    {
```

```
        for(j=0;j<c;j++)
```

```
        {
```

```
            scanf("%d", &m[i][j]);
```

```
        }
```

```
    }
```

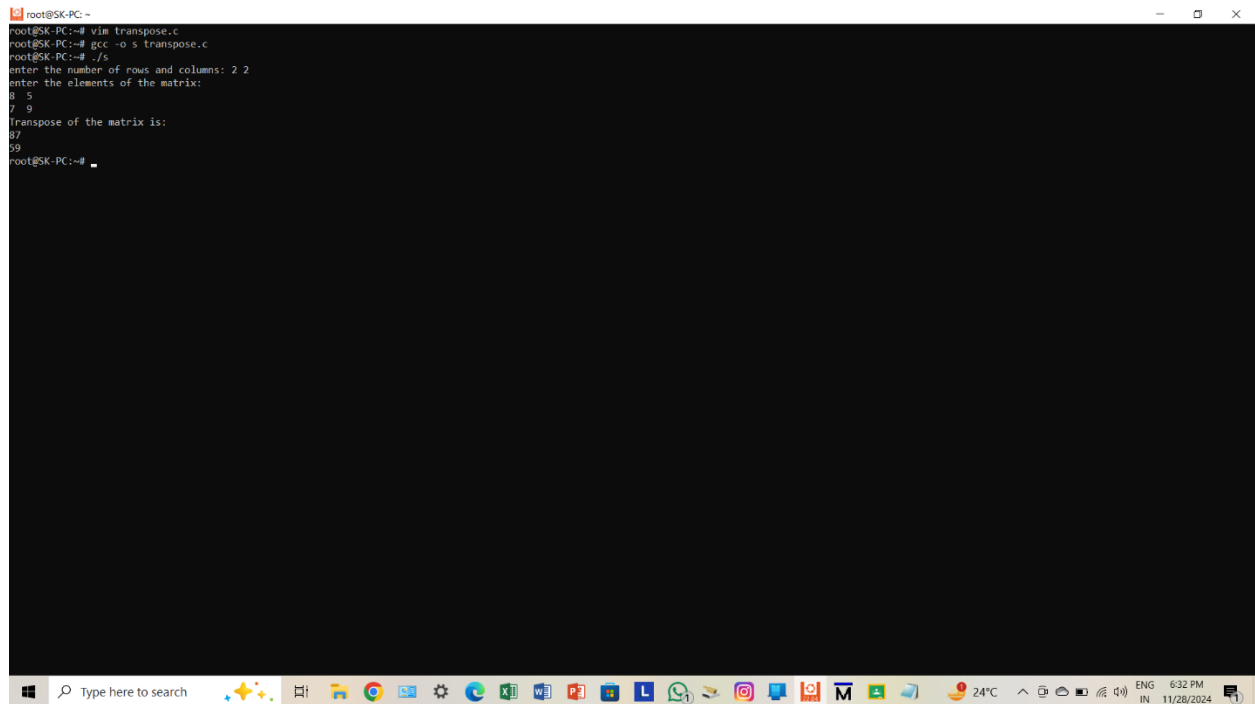
```
    for(i=0;i<r;i++)
```

```
    {
```

```
        for(j=0;j<c;j++)
```

```
        {  
            t[j][i]=m[i][j];  
        }  
    }  
    printf("Transpose of the matrix is: \n");  
    for(i=0;i<c;i++)  
    {  
        for(j=0;j<r;j++)  
        {  
            printf("%d", t[i][j]);  
        }  
        printf("\n");  
    }  
  
    return 0;  
}
```

OUTPUT



```
root@SK-PC: ~  
root@SK-PC:~# vim transpose.c  
root@SK-PC:~# gcc -o s transpose.c  
root@SK-PC:~# ./s  
enter the number of rows and columns: 2 2  
enter the elements of the matrix:  
8 5  
7 9  
Transpose of the matrix is:  
87  
59  
root@SK-PC:~#
```

The screenshot shows a terminal window with a black background and white text. The user is at a prompt 'root@SK-PC: ~'. They run 'vim transpose.c', then 'gcc -o s transpose.c', and finally './s'. The program prompts for 'enter the number of rows and columns: 2 2' and 'enter the elements of the matrix:'. The user enters '8 5' and '7 9'. The program outputs 'Transpose of the matrix is:', followed by '87' and '59' on separate lines. The terminal window has a standard Linux-style title bar and a Windows taskbar at the bottom with various icons and system information like '24°C' and '6:32 PM 11/28/2024'.

14. Write a program to add the elements of each row and each column of a matrix.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int r,c,i,j;
```

```
    printf("enter the number of rows and columns: ");
```

```
    scanf("%d %d", &r, &c);
```

```
    int mat[r][c];
```

```
printf("enter the elements of the matrix:\n");
```

```
for(i=0;i<r;i++)
```

```
{
```

```
    for(j=0;j<c;j++)
```

```
    {
```

```
        scanf("%d", &mat[i][j]);
```

```
    }
```

```
}
```

```
printf("Sum of each row:\n");
```

```
for(i=0;i<r;i++)
```

```
{
```

```
    int rowsum=0;
```

```
    for(j=0;j<c;j++)
```

```
    {
```

```
        rowsum=rowsum+mat[i][j];
```

```
    }
```

```
    printf("Row %d: %d\n",i+1,rowsum);
```

```
}
```

```
printf("Sum of each column:\n");
```

```
for(i=0;i<r;i++)
```

```
{
```

```
int colsum=0;

for(j=0;j<c;j++)

{

    colsum=colsum+mat[i][j];

}

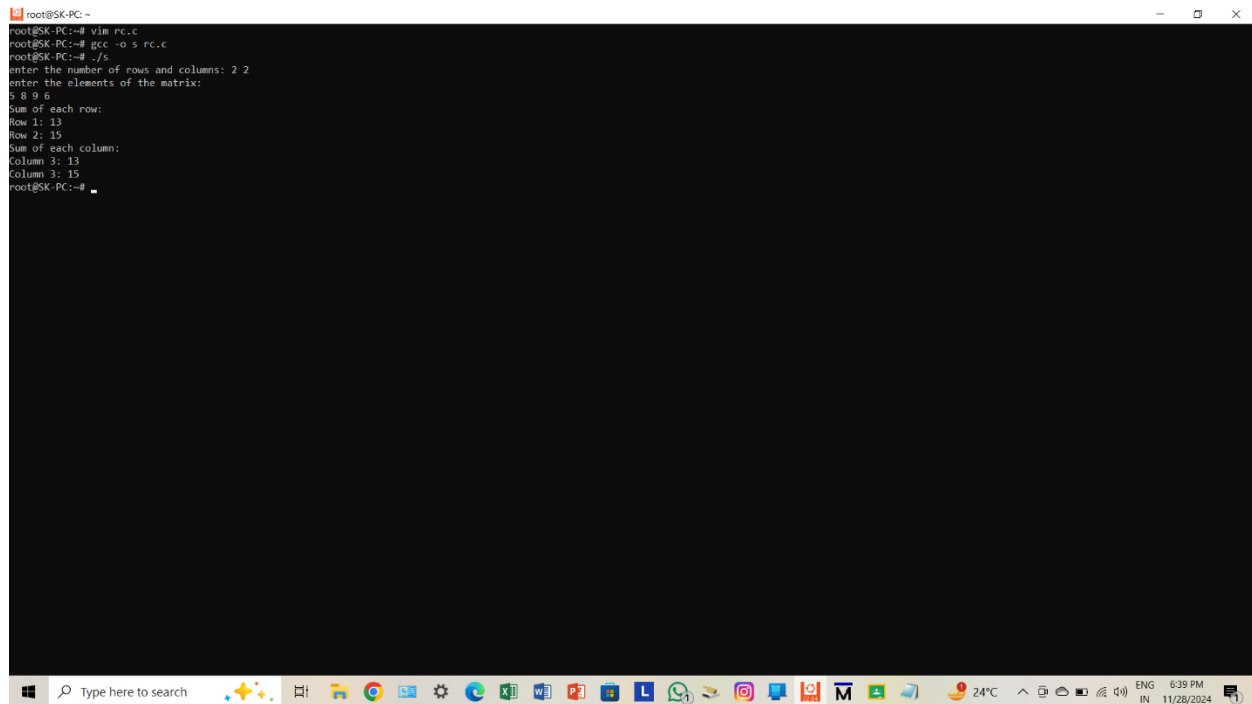
printf("Column %d: %d\n",j+1,colsum);

}

return 0;

}
```

OUTPUT



```
root@SK-PC:~#  
root@SK-PC:~# vim rc.c  
root@SK-PC:~# gcc -o 5 rc.c  
root@SK-PC:~# ./5  
enter the number of rows and columns: 2 2  
enter the elements of the matrix:  
5 8 9 6  
Sum of each row:  
Row 1: 13  
Row 2: 15  
Sum of each column:  
Column 1: 13  
Column 2: 15  
root@SK-PC:~#
```


15. Write a program to perform the multiplication of two matrices.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int r,c,r1,c1,i,j,k;
```

```
    printf("enter the number of rows and columns of the first matrix:");
```

```
    scanf("%d %d", &r, &c);
```

```
    printf("enter the number of rows and columns of the second matrix:");
```

```
    scanf("%d %d", &r1, &c1);
```

```
    if(c!=r1)
```

```
    {
```

```
        printf("matrix multiplication is not possible.\n");
```

```
        return 1;
```

```
    }
```

```
    int m1[r][c], m2[r1][c1], result[r][c1];
```

```
printf("enter the elements of the first matrix:\n");
```

```
for(i=0;i<r;i++)
```

```
{
```

```
    for(j=0;j<c;j++)
```

```
    {
```

```
        scanf("%d", &m1[i][j]);
```

```
    }
```

```
}
```

```
printf("enter the elements of the second matrix:\n");
```

```
for(i=0;i<r1;i++)
```

```
{
```

```
    for(j=0;j<c1;j++)
```

```
    {
```

```
        scanf("%d", &m2[i][j]);
```

```
    }
```

```
}
```

```
for(i=0;i<r;i++)
```

```
{
```

```
    for(j=0;j<c1;j++)
```

```
    {  
        result[i][j]=0;  
    }  
}
```

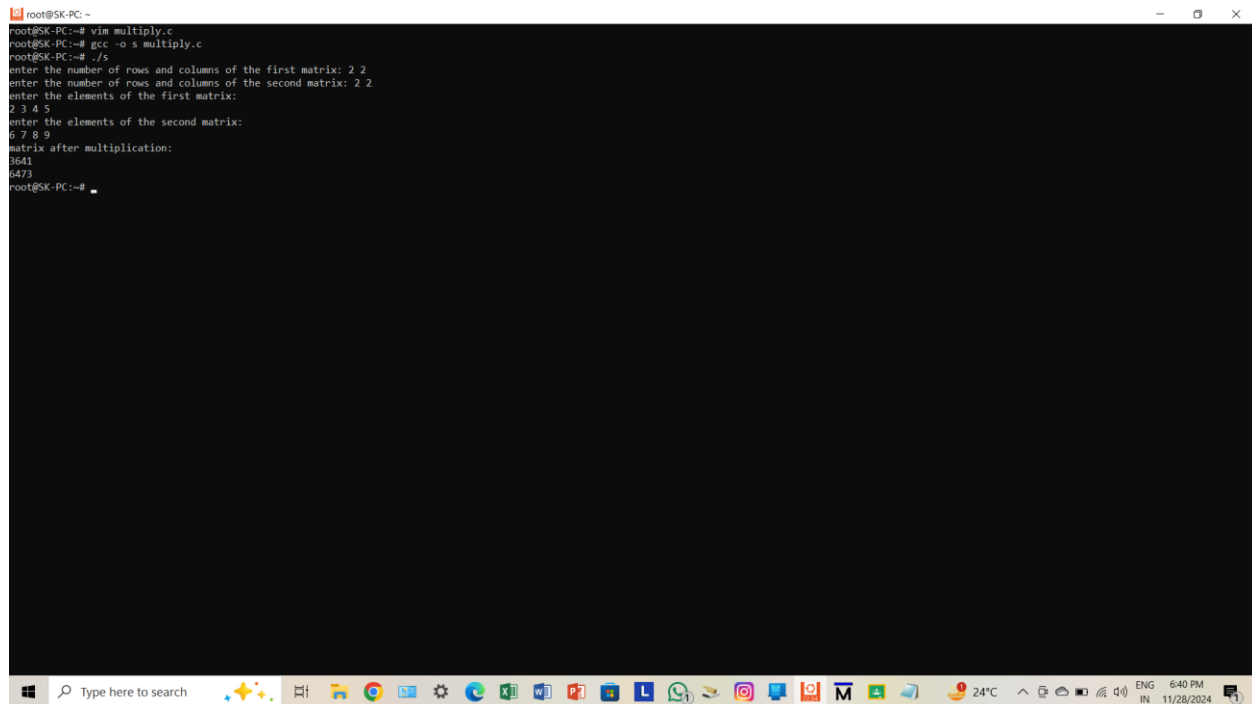
```
for(i=0;i<r;i++)  
{  
    for(j=0;j<c1;j++)  
    {  
        for(k=0;k<c;k++)  
        {  
            result[i][j] += m1[i][k] * m2[k][j];  
        }  
    }  
}
```

```
printf("matrix after multiplication:\n");
```

```
for(i=0;i<r;i++)  
{  
    for(j=0;j<c1;j++)  
    {
```

```
        printf("%d", result[i][j]);  
  
    }  
  
    printf("\n");  
  
}  
  
return 0;  
  
}
```

OUTPUT



```
root@SK-PC: ~  
root@SK-PC:~# vim multiply.c  
root@SK-PC:~# gcc -o s multiply.c  
root@SK-PC:~# ./s  
enter the number of rows and columns of the first matrix: 2 2  
enter the number of rows and columns of the second matrix: 2 2  
enter the elements of the first matrix:  
2 3 4 5  
enter the elements of the second matrix:  
6 7 8 9  
matrix after multiplication:  
3641  
6473  
root@SK-PC:~#
```

The screenshot shows a terminal window with a black background and white text. The user is in a directory named '~' on a system named 'SK-PC'. They have opened a file 'multiply.c' in the 'vim' editor, compiled it with 'gcc -o s multiply.c', and then executed the resulting binary 's'. The program prompts for the dimensions and elements of two matrices. The first matrix is 2x2 with elements [2, 3, 4, 5] and the second matrix is 2x2 with elements [6, 7, 8, 9]. The output shows the result of the multiplication as two rows: '3641' and '6473'. The Windows taskbar is visible at the bottom of the screen.

16. Write a program to check whether a matrix is an identity matrix or not.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int n,i,j,isIdentity=1;
```

```
    printf("enter the size of the square matrix (n x n): ");
```

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

```
    int m[n][n];
```

```
    printf("enter the elements of the matrix:\n");
```

```
    for(i=0;i<n;i++)
```

```
    {
```

```
        for(j=0;j<n;j++)
```

```
        {
```

```
            scanf("%d", &m[i][j]);
```

```
        }
```

```
    }
```

```
    for(i=0;i<n;i++)
```

```
    {
```

```
        for(j=0;j<n;j++)
```

```
{
    if((i==j && m[i][j]!=1) || (i!=j && m[i][j]!=0))
    {
        isIdentity=0;
        break;
    }
}

if(isIdentity)
{
    printf("The matrix is an identity matrix.\n");
}
else
{
    printf("The matrix is not an identity matrix.\n");
}

return 0;
}
```

OUTPUT

```
root@SK-PC: ~  
root@SK-PC:~# vim identity.c  
root@SK-PC:~# gcc -o s identity.c  
root@SK-PC:~# ./s  
enter the size of the square matrix (n x n): 3 3  
enter the elements of the matrix:  
1 0 0  
0 1 0  
0 0 1  
The matrix is not an identity matrix.  
root@SK-PC:~#
```

17. Write a program to check whether a matrix is a sparse matrix or not.

```
int main()  
{  
    int r,c,i,j,zeroCount=0;  
    printf("enter the number of rows and columns:");  
    scanf("%d %d", &r, &c);  
    int arr[r][c];  
    printf("enter the elements:\n");  
    for(i=0;i<r;i++)
```

```
{  
    for(j=0;j<c;j++)  
    {  
        scanf("%d", &arr[i][j]);  
        if(arr[i][j]==0)  
        {  
            zeroCount++;  
        }  
    }  
}  
if(zeroCount > (r*c)/2)  
{  
    printf("The matrix is a sparse matrix.\n");  
}  
else  
{  
    printf("The matrix is not a sparse matrix.\n");  
}  
  
return 0;  
}
```


OUTPUT

```
root@SK-PC: ~  
root@SK-PC:~# vim sparse.c  
root@SK-PC:~# gcc -o s sparse.c  
root@SK-PC:~# ./s  
enter the number of rows and columns: 4 5  
enter the elements:  
0 0 3 0 4  
0 0 5 7 0  
0 0 0 0 0  
0 2 6 0 0  
The matrix is a sparse matrix.  
root@SK-PC:~#
```

18. Write a C program to create a structure named company which has name, address, phone and no Of Employee as member variables. Read the name of the company, its address, phone and no Of Employee. Finally display these members" values.

```
#include <stdio.h>
```

```
struct company
```

```
{
```

```
    char name[50];
```

```
    char address[100];
```

```
    char phone[15];
```

```
    int employee;
```

```
};
```

```
int main()
{
    struct company c;

    printf("enter the company name: ");
    fgets(c.name, sizeof(c.name), stdin);

    printf("enter the address: ");
    fgets(c.address, sizeof(c.address), stdin);

    printf("enter the phone: ");
    fgets(c.phone, sizeof(c.phone), stdin);

    printf("enter the number of employees: ");
    scanf("%d", &c.employee);


    printf("\nComapny Details:\n");
    printf("Name: %s", c.name);
    printf("Address: %s", c.address);
    printf("Phone: %s", c.phone);
    printf("Number of employees: %d\n", c.employee);

    return 0;
}
```

OUTPUT

```
root@SK-PC: ~  
root@SK-PC:~# vim structure.c  
root@SK-PC:~# gcc -o s structure.c  
root@SK-PC:~# ./s  
enter the company name: Mahindra Motors  
enter the address: New Dehli  
enter the phone: 9784464615  
enter the number of employees: 1000  
  
Company Details:  
Name: Mahindra Motors  
Address: New Dehli  
Phone: 9784464615  
Number of employees: 1000  
root@SK-PC:~#
```

19. Define a structure “complex” (typedef) to read two complex numbers and perform addition and subtraction of these two complex numbers and display the result.

```
#include <stdio.h>
```

```
typedef struct complex
```

```
{
```

```
    float real;
```

```
    float imag;
```

```
} Complex;
```

```
Complex add(Complex c1, Complex c2)
```

```
{  
    Complex result;  
    result.real=c1.real+c2.real;  
    result.imag=c1.imag+c2.imag;  
    return result;  
}
```

```
Complex sub(Complex c1, Complex c2)
```

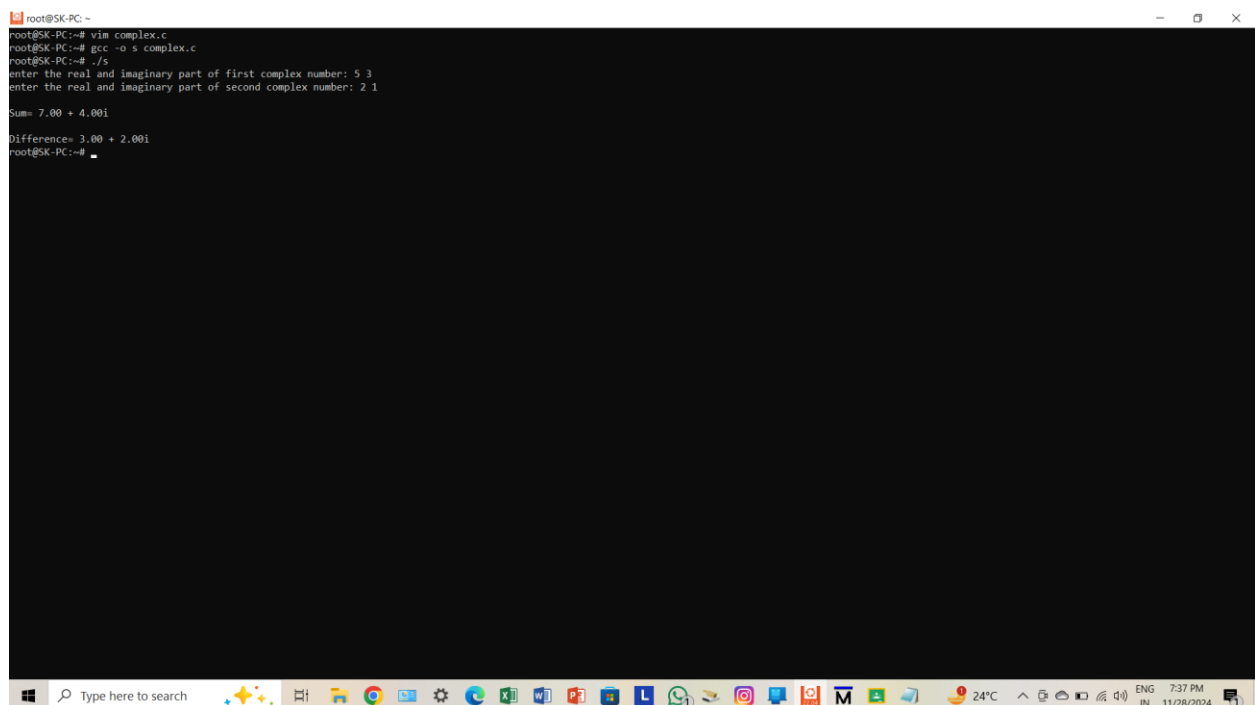
```
{  
    Complex result;  
    result.real=c1.real-c2.real;  
    result.imag=c1.imag-c2.imag;  
    return result;  
}
```

```
int main()
```

```
{  
    Complex c1,c2,sum,diff;  
    printf("enter the real and imaginary part of first complex number: ");  
    scanf("%f %f", &c1.real, &c1.imag);
```

```
printf("enter the real and imaginary part of second complex number: ");  
  
scanf("%f %f", &c2.real, &c2.imag);  
  
sum= add(c1, c2);  
  
diff= sub(c1, c2);  
  
printf("\nSum= %.2f + %.2fi\n", sum.real, sum.imag);  
  
printf("\nDifference= %.2f + %.2fi\n", diff.real, diff.imag);  
  
return 0;  
  
}
```

OUTPUT



```
root@SK-PC: ~  
root@SK-PC:~# vim complex.c  
root@SK-PC:~# gcc -o s complex.c  
root@SK-PC:~# ./s  
enter the real and imaginary part of first complex number: 5 3  
enter the real and imaginary part of second complex number: 2 1  
  
Sum= 7.00 + 4.00i  
  
Difference= 3.00 + 2.00i  
root@SK-PC:~#
```

The screenshot shows a terminal window with a black background and white text. The user is in a directory named '~' on a machine named 'root@SK-PC'. They have edited a file 'complex.c' using 'vim', compiled it with 'gcc -o s complex.c', and executed it with './s'. The program prompts for the real and imaginary parts of two complex numbers. The first number is 5 + 3i and the second is 2 + 1i. The program outputs the sum as 7.00 + 4.00i and the difference as 3.00 + 2.00i. The Windows taskbar is visible at the bottom, showing the time as 7:37 PM on 11/28/2024.

20. Write a C program to read the RollNo, Name, Address, and Age Marks of 12 students in the BCT class and display the details from the function.

```
#include <stdio.h>
```

```
struct student
```

```
{
```

```
    int rollNo;
```

```
    char name[50];
```

```
    char address[100];
```

```
    int age;
```

```
    float marks;
```

```
};
```

```
void display(struct student students[], int n)
```

```
{
```

```
    printf("\nStudent Details:\n");
```

```
    for(int i=0;i<n;i++)
```

```
    {
```

```
        printf("Student %d:\n", i+1);
```

```
        printf("Roll no: %d\n", students[i].rollNo);
```

```
        printf("Name: %s", students[i].name);
```

```
        printf("Address: %s", students[i].address);
```

```
        printf("Age: %d\n", students[i].age);
```

```
        printf("Marks: %.2f\n", students[i].marks);
    }
}

int main()
{
    struct student students[12];
    int n=12;

    for(int i=0;i<n;i++)
    {
        printf("enter the details of student %d:\n", i+1);
        printf("Roll No: ");
        scanf("%d", &students[i].rollNo);
        getchar();
        printf("Name: ");
        fgets(students[i].name, sizeof(students[i].name), stdin);
        printf("Address: ");
        fgets(students[i].address, sizeof(students[i].address), stdin);
        printf("Age: ");
        scanf("%d", &students[i].age);
        printf("Marks: ");
        scanf("%f", &students[i].marks);
    }
}
```

```
    display(students, n);  
    return 0;  
}
```