Name	**Shubhan Singh**
UID no.	**2022300118**
Experiment No.	**6**

AIM:	**To demonstrate the use of two-dimensional arrays to solve a given problem.**
Program 1	
PROBLEM STATEMENT :	*Write a program to perform Matrix Addition, Subtraction, Multiplication, Transpose of Matrix and Norm of Matrix. Dimensions of matrices will be decided by the user.*
ALGORITHM:	Algorithm for the **main()** function: Step 1: START Step 2: Read the dimensions of two matrices from user input. Step 3: Declare two matrices with sizes as specified from input. Step 4: Populate both the matrices with elements read from user input. Step 5: if the dimensions of both the matrices are same, execute print_sum_matrix function, passing both matrices and their dimensions into the function. Step 6: else print that the sum of these matrices doesn't exist. Step 7: if the dimensions of both the matrices are same, execute print_diff_matrix function, passing both matrices and their dimensions into the function. Step 8: else print that the difference of these matrices doesn't exist. Step 9: if horizontal dimension of one matrix is equal to the vertical dimension of other matrix, execute print_prod_matrix, passing both matrices and their dimensions into the function. Step 10: else print that product of these matrices doesn't exist. Step 11: Execute the print_transpose_matrix function for both the matrices respectively, passing the matrices and their dimensions into the function. step 12: Print the norm of both the matrices using the print_norm_matrix function for both the matrices respectively, passing the matrices and their dimensions into the function. step 13: END Algorithm for the **print_sum_matrix(int y, int x, int matrix1[y][x], int a, int b, int matrix2[a][b])** function: Step 1: Declare a 2-d array summatrix with the same dimensions as the passed arrays. Step 2: initialise a variable i to 0 Step 3: initialise a variable j to 0

	Step 4: set the element summmatrix[i][j] to matrix1[i][j]+matrix2[i][j] Step 5: increment j. Step 6: if j<x, return to step 4. Step 7: increment i. Step 8: if i<y, return to step 3. Step 9: Print the array summatrix using the function print_matrix, by passing summatrix and its dimensions into the function. Algorithm for the **print_diff_matrix(int y, int x, int matrix1[y][x], int a, int b, int matrix2[a][b])** function: Step 1: Declare a 2-d array diffmatrix with the same dimensions as the passed arrays. Step 2: initialize a variable i to 0 Step 3: initialize a variable j to 0 Step 4: set the element diffmatrix[i][j] to matrix1[i][j]-matrix2[i][j] Step 5: increment j. Step 6: if j<x, return to step 4. Step 7: increment i. Step 8: if i<y, return to step 3. Step 9: Print the array diffmatrix using the function print_matrix, by passing diffmatrix and its dimensions into the function. Algorithm for the **print_prod_matrix(int y, int x, int matrix1[y][x], int a, int b, int matrix2[a][b])** function: Step 1: Declare a 2-d array prodmatrix with the dimensions y(vertical) x b(horizontal). Step 2: initialize all elements of prodmatrix to 0. Step 3: initialize i=0 Step 4: initialize j=0 Step 5: initialize p=0 Step 6: set prodmatrix[i][j] = prodmatrix[i][j]+matrix1[i][p]*matrix2[p][j] Step 7: increment p Step 8: if p<x, return to step 6 Step 9: if j<b, return to step 5 Step 10: if i<y, return to step 4 Step 11: print prodmatrix using the function print_matrix, by passing prodmatrix and its dimensions into the function. Algorithm for the function **print_transpose_matrix(int *y*,int *x*,int *matrix*[*y*][*x*])** : Step 1: declare a matrix transpose with the vertical dimension equal to the horizontal dimension of the passed matrix and the horizontal dimension equal to vertical dimension of passed matrix.

	Step 2: initialise a variable i to 0 Step 3: initialise a variable j to 0 Step 4: set the element transpose[j][i]=matrix[i][j] Step 5: increment j. Step 6: if j<x, return to step 4. Step 7: increment i. Step 8: if i<y, return to step 3. Step 9: print transpose using the function print_matrix, by passing transpose and its dimensions into the function. Algorithm for the **int print_norm_matrix(int *y*,int *x*, int *matrix*[*y*][*x*])** function: Step 1: initialize two variables norm and maxnorm to 0. Step 2: initialise a variable i to 0 Step 3: initialise a variable j to 0 Step 4: norm=norm+matrix[j][i] Step 5: increment j. Step 6: if j<y, return to step 4. Step 7: if i=0, i.e. in the first iteration, set maxnorm=norm, in subsequent iterations, if norm>maxnorm, set maxnorm=norm. Step 8: increment i. Step 9: if i<x, return to step 3. Step 10: return maxnorm. Algorithm for the **void print_matrix(int *y*,int *x*,int *matrix*[*y*][*x*])** function: (I have ignored the steps for formatting the output here) Step 1: initialize a variable i to 0 Step 2: initialize a variable j to 0 Step 3: print the value of matrix[i][j] Step 4: increment j. Step 5: if j<x, return to step 3. Step 6: print newline Step 7: increment i. Step 8: if i<y, return to step 2.

PROGRAM:

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

void print_matrix(int y,int x,int matrix[y][x]){
    int max=matrix[0][0];
    int maxdigits=0;
    for(int i=0;i<y;i++){
        for(int j=0;j<x;j++){
            if(matrix[i][j]>max){max=matrix[i][j];}
        }
    }
    while(max>0){
        maxdigits++;
        max=max/10;
    }
    for(int i=0;i<y;i++){
        for(int j=0;j<x;j++){
            printf("| %*d |",maxdigits+1,matrix[i][j]);
        }
        printf("\n");
    }
}
void print_sum_matrix(int y,int x,int matrix1[y][x],int a, int b,int
matrix2[a][b]){
    int summatrix[y][x];
    for(int i=0;i<y;i++){
        for(int j=0;j<x;j++){
            summatrix[i][j]=matrix1[i][j]+matrix2[i][j];
        }
    }
    print_matrix(y,x,summatrix);
}
void print_diff_matrix(int y,int x, int matrix1[y][x],int a,int b, int
matrix2[a][b]){
    int diffmatrix[y][x];
    for(int i=0;i<y;i++){
        for(int j=0;j<x;j++){
            diffmatrix[i][j]=matrix1[i][j]-matrix2[i][j];
        }
    }
    print_matrix(y,x,diffmatrix);
}
void print_prod_matrix(int y,int x, int matrix1[y][x],int a,int b, int
matrix2[a][b]){
```

```
    int prodmatrix[y][b];
    for(int i=0;i<y;i++){
        for(int j=0;j<b;j++){
            prodmatrix[i][j]=0;
        }
    }
    for(int i=0;i<y;i++){
        for(int j=0;j<b;j++){
            for(int p=0;p<x;p++){
                prodmatrix[i][j]=prodmatrix[i][j]+matrix1[i][p]*matrix2[p][j];
            }
        }
    }
    print_matrix(y,b,prodmatrix);
}
void print_transpose_matrix(int y,int x,int matrix[y][x]){
    int transpose[x][y];
    for(int i=0;i<y;i++){
        for(int j=0;j<x;j++){
            transpose[j][i]=matrix[i][j];
        }
    }
    print_matrix(x,y,transpose);
    printf("\n");
}
int print_norm_matrix(int y,int x, int matrix[y][x]){
    int norm=0,maxnorm=0;
    for(int i=0;i<x;i++){
        for(int j=0;j<y;j++){
            norm=norm+matrix[j][i];
        }
        if(i==0){maxnorm=norm;}
        if(norm>maxnorm){maxnorm=norm;}
    }
    return maxnorm;
}
int main(){
    int y,x,a,b,n1,n2;
    printf("Enter vertical dimensions of first matrix\n");
    scanf("%d",&y);
    printf("Enter horizontal dimensions of first matrix\n");
    scanf("%d",&x);
    printf("Enter vertical dimensions of second matrix\n");
    scanf("%d",&a);
```

```
    printf("Enter horizontal dimensions of second matrix\n");
    scanf("%d",&b);
    int matrix1[y][x];
    int matrix2[a][b];
    printf("Enter the elements of the first matrix(row by row, from left to
right)\n");
    for(int i=0;i<y;i++){
        for(int j=0;j<x;j++){
            scanf("%d",&matrix1[i][j]);
        }
    }
    printf("Enter the elements of the second matrix(row by row, from left to
right)\n");
    for(int i=0;i<a;i++){
        for(int j=0;j<b;j++){
            scanf("%d",&matrix2[i][j]);
        }
    }
    if((y==a)&&(x==b)){
    printf("The sum of the two matrices is:\n");
    print_sum_matrix(y,x,matrix1,a,b,matrix2);
    }
    else{printf("The sum of these matrices doesnt exist\n");}
    if((y==a)&&(x==b)){
    printf("The difference of the two matrices is:\n");
    print_diff_matrix(y,x,matrix1,a,b,matrix2);
    }
    else{printf("The difference of these matrices doesnt exist\n");}
    if(x==a){
    printf("The product of the two matrices is:\n");
    print_prod_matrix(y,x,matrix1,a,b,matrix2);
    }
    else{printf("The product of these matrices doesnt exist\n");}
    printf("The transpose of the two matrices is:\n");
    print_transpose_matrix(y,x,matrix1);
    print_transpose_matrix(a,b,matrix2);
    n1=print_norm_matrix(y,x,matrix1);
    printf("The norm of matrix 1 is: %d\n",n1);
    n2=print_norm_matrix(a,b,matrix2);
    printf("The norm of matrix 2 is: %d\n",n2);
    return 0;
}
```

RESULT:

```
Enter vertical dimensions of first matrix
3
Enter horizontal dimensions of first matrix
3
Enter vertical dimensions of second matrix
3
Enter horizontal dimensions of second matrix
3
Enter the elements of the first matrix(row by row, from left to right)
324 -54 76
98 -12 76
-97 128 653
Enter the elements of the second matrix(row by row, from left to right)
123 786 324
23 -99 32
12 98 2
The sum of the two matrices is:
|  447 ||  732 ||  400 |
|  121 || -111 ||  108 |
|  -85 ||  226 ||  655 |
The difference of the two matrices is:
|  201 || -840 || -248 |
|   75 ||   87 ||   44 |
| -109 ||   30 ||  651 |
The product of the two matrices is:
|   39522 ||  267458 ||  103400 |
|   12690 ||   85664 ||   31520 |
|   -1151 ||  -24920 ||  -26026 |
The transpose of the two matrices is:
|  324 ||   98 ||  -97 |
|  -54 ||  -12 ||  128 |
|   76 ||   76 ||  653 |

|  123 ||   23 ||   12 |
|  786 ||  -99 ||   98 |
|  324 ||   32 ||    2 |

The norm of matrix 1 is: 1192
The norm of matrix 2 is: 1301
```

```
Enter vertical dimensions of first matrix
4
Enter horizontal dimensions of first matrix
2
Enter vertical dimensions of second matrix
2
Enter horizontal dimensions of second matrix
3
Enter the elements of the first matrix(row by row, from left to right)
45 22
-81 11
73 955
63 53
Enter the elements of the second matrix(row by row, from left to right)
11 33 11
76 54 -32
The sum of these matrices doesnt exist
The difference of these matrices doesnt exist
The product of the two matrices is:
|   2167 ||   2673 ||   -209 |
|    -55 ||  -2079 ||  -1243 |
|  73383 ||  53979 || -29757 |
|   4721 ||   4941 ||  -1003 |
The transpose of the two matrices is:
|   45 ||  -81 ||   73 ||   63 |
|   22 ||   11 ||  955 ||   53 |

|  11 ||  76 |
|  33 ||  54 |
|  11 || -32 |

The norm of matrix 1 is: 1141
The norm of matrix 2 is: 174
```

Program 2	
PROBLEM STATEMENT :	*Write a program which reads the current year followed by N followed by a list of N employee numbers and their current ages. Produce a list showing the years in which the employees retire (become 65 years old). If more than one employee retires in a given year then include them all under the same heading.*
ALGORITHM:	Step 1: START Step 2: Read current year(y) and number of employees(N) from input. Step 3: Initialize an integer array arremployee of dimensions N x 3 Step 3: Read the employee numbers of N employees and their ages from input Step 4: Store the employee numbers in the array locations with indexes [0][0] to [N-1][0] and their ages in the locations [0][1] to [N-1][1] respectively. Step 5: Store the retirement age of employees in the array locations [0][2] to [N-1][2] in the

	following manner: arremployee[i][2]=y-arremployee[i][1]+65, with 65 being the retirement age. Step 6: sort the employees according to their retirement ages, i.e. sort the rows in the 2d array based upon the 3rd element in each row. (Sorting algorithm used is insertion sort) Step 7: initialize k=0 Step 8: set flag=arremployee[k][2] Step 9: print “age of retirement is” followed by value in arremployee[k][2] Step 10: print “list of employees: ” Step 11: print value of arremployee[k][0] Step 12: increment k Step 13: if arremployee[k][2] equals flag, return to step 11 Step 14: print newline Step 15: if k<N, return to step 8 Step 16: END

PROGRAM:	`#include<stdio.h>`
	`void sort (int n,int m, int arr[n][m]){`
	`  int temp,j,temp2,temp3;`
	`  for(int i=0;i<n-1;i++){`
	`  j=i+1;`
	`  while(arr[j-1][2]>arr[j][2]){`
	`  temp=arr[j-1][2];`
	`  arr[j-1][2]=arr[j][2];`
	`  arr[j][2]=temp;`
	`  temp2=arr[j-1][1];`
	`  arr[j-1][1]=arr[j][1];`
	`  arr[j][1]=temp2;`
	`  temp3=arr[j-1][0];`
	`  arr[j-1][0]=arr[j][0];`
	`  arr[j][0]=temp3;`
	`  j--;`
	`  if(j==0){break;}`
	`  }`
	`  }`
	`}`
	`int main(){`
	`  int y,N,flag,k=0,m=3;`
	`  printf("Enter current year: ");`
	`  scanf("%d",&y);`
	`  printf("\nEnter number of employees(N): ");`
	`  scanf("%d",&N);`
	`  int arremployee[N][3];`

```
    printf("\nenter %d sets of employee numbers and ages(seperated by
spaces):\n",N);
    for(int i=0;i<N;i++){
        scanf("%d",&arremployee[i][0]);
        scanf("%d",&arremployee[i][1]);
    }
    for(int i=0;i<N;i++){
        arremployee[i][2]=y-arremployee[i][1]+65;
    }
    sort(N,m,arremployee);
    while(k<N){
        flag=arremployee[k][2];
        printf("Year of retirement: %d\n",arremployee[k][2]);
        printf("List of employees: ");
        while(arremployee[k][2]==flag){
        printf("%d   ",arremployee[k][0]);
        k++;
        }
        printf("\n");
    }
    return 0;
}
```

RESULT:

```
Enter current year: 2023

Enter number of employees(N): 8

enter 8 sets of employee numbers and ages(seperated by spaces):
123001 45
123002 54
123003 19
123004 25
123005 21
123006 49
123007 40
123008 29
Year of retirement: 2034
List of employees: 123002
Year of retirement: 2039
List of employees: 123006
Year of retirement: 2043
List of employees: 123001
Year of retirement: 2048
List of employees: 123007
Year of retirement: 2059
List of employees: 123008
Year of retirement: 2063
List of employees: 123004
Year of retirement: 2067
List of employees: 123005
Year of retirement: 2069
List of employees: 123003
```

```
Enter current year: 2023

Enter number of employees(N): 15

enter 15 sets of employee numbers and ages(seperated by spaces):
123001 45
123004 43
123007 34
123008 43
123017 23
123465 22
123487 54
123654 39
123408 23
123543 30
123670 55
123444 39
123764 45
123033 55
123961 45
Year of retirement: 2033
List of employees: 123670   123033
Year of retirement: 2034
List of employees: 123487
Year of retirement: 2043
List of employees: 123001   123764   123961
Year of retirement: 2045
List of employees: 123004   123008
Year of retirement: 2049
List of employees: 123654   123444
Year of retirement: 2054
List of employees: 123007
Year of retirement: 2058
List of employees: 123543
Year of retirement: 2065
List of employees: 123017   123408
Year of retirement: 2066
List of employees: 123465
```

CONCLUSION:	We learnt how to use 2 dimensional arrays in problem solving using computer programming.