Assignment 2

1. In a class of 50 students four courses were taught in a given semester namely Maths, Science, English, and Hindi respectively. They have a strict grading system with the following

range of marks, 'A': 90-100, 'B': 75-89, 'C': 60-74, 'D': 45-59, 'E': 33-44, and F: 0-32.

Given course instructor already evaluates the answer sheets of the students, Write a C

program to solve the following issues

(a) How many students have got the grade 'A' in Maths?

(b) Find the highest and lowest marks obtained by a student in English and Hindi respectively.

(c) How many students failed in the semester, given that the student fails if they got a

grade 'F' in more than three subjects?

(d) Find out the overall average of class, and a subject-wise average of class

(e) Evaluate the percentage of each student.

CODE

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int **matrix(int m,int n)
{
    int **p=(int**)calloc(m,sizeof(int*));
    for(int i=0; i<m; i++)
    {
        p[i]=(int*)calloc(n,sizeof(int));
    }
    return p;
}
void subject_marks(int *p)
{
    int lowest = 0, highest = 100;
    for (int i = 0; i < 4; i++)
    {
        p[i] = (rand() % (highest - lowest + 1)) + lowest;
    }

}

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

```
        printf("\n");
    }
}
int maths(int **p,int m,int n)
{
    int c=0;
    for(int i=0; i<m; i++)
    {
        if(p[i][0]>=90)
        {
            c=c+1;
        }
    }
    return c;

}
int highest(int **p,int m,char subject[10])
{
    int mx;
    if(strcmp(subject,"english")==0)
    {
        mx=p[0][3];
        for(int i=1; i<m; i++)
        {
            if(p[i][2]>mx)
            {
                mx=p[i][2];
            }
        }
        return mx;


    }
    if(strcmp(subject,"hindi")==0)
    {
        int mx=p[0][3];
        for(int i=1; i<m; i++)
        {
            if(p[i][3]>mx)
            {
                mx=p[i][3];

            }
        }
        return mx;


    }

}
int lowest(int **p,int m,char subject[20])
{
```

```
    int mn;
    if(strcmp(subject,"english")==0)
    {
        mn=p[0][2];
        for(int i=1; i<m; i++)
        {
            if(p[i][2]<mn)
            {
                mn=p[i][2];
            }
        }
        return mn;


    }
    if(strcmp(subject,"hindi")==0)
    {
        int mn=p[0][3];
        for(int i=0; i<m; i++)
        {
            if(p[i][3]<mn)
            {
                mn=p[i][3];

            }
        }
        return mn;


    }

}
int fail(int **p,int stud_no)
{
    int s=0;
    for(int i=0; i<stud_no; i++)
    {
        int c=0;
        for(int j=0; j<4; j++)
        {
            if(p[i][j]<=32)
            {

                c=c+1;
            }

        }
        if(c==4)
        {
            s=s+1;
        }
    }
```

```
    return s;
}
float avg(int **p,int stud_no)
{
    float s=0;
    for(int i=0; i<stud_no; i++)
    {
        for(int j=0; j<4; j++)
        {
            s=s+p[i][j];

        }

    }
    s=s/200;
    return s;

}
float sub_avg(int **p,int m,char subject[20])
{
    float s=0;
    if(strcmp(subject,"maths")==0)
    {
        for(int i=0; i<m; i++)
        {
            s=s+p[i][0];
        }
        return s/50;


    }
    if(strcmp(subject,"science")==0)
    {

        for(int i=0; i<m; i++)
        {
            s=s+p[i][1];
        }
        return s/50;


    }
    if(strcmp(subject,"english")==0)
    {

        for(int i=0; i<m; i++)
        {
            s=s+p[i][2];
        }
        return s/50;


    }
```

```
    if(strcmp(subject,"hindi")==0)
    {

        for(int i=0; i<m; i++)
        {
            s=s+p[i][3];
        }
        return s/50;


    }


}
float percent(int *p)
{
    float s=0.0;
    for(int i=0; i<4; i++)
    {
        s=s+p[i];
    }
    float per=(s*100.0)/400.0;
    return per;
}
int main()
{
    int stud_no=50,subject=4;
    int **marks=matrix(50,4);
    for(int i=0; i<stud_no; i++)
    {
        subject_marks(marks[i]);

    }
    display(marks,50,4);
    printf("%d students have got grade A in Maths\n",maths(marks,50,4));
    printf("highest marks in english is %d and lowest marks in english is
%d\n",highest(marks,50,"english"),lowest(marks,50,"english"));
    printf("highest marks in hindi is %d and lowest marks in hindi is
%d\n",highest(marks,50,"hindi"),lowest(marks,50,"hindi"));
    printf("No of students failed:%d\n",fail(marks,stud_no));
    printf("overall average:%f\n",avg(marks,stud_no));
    printf("average marks of maths: %f\n",sub_avg(marks,stud_no,"maths"));
    printf("average marks of science: %f\n",sub_avg(marks,stud_no,"science"));
    printf("average marks of english: %f\n",sub_avg(marks,stud_no,"english"));
    printf("average marks of hindi: %f\n",sub_avg(marks,stud_no,"hindi"));
    for(int i=0; i<stud_no; i++)
    {
        printf("percentage of student %d is: %f\n",i+1,percent(marks[i]));
    }

}
```

2. Given n users (U) and m movies (M) with their rating represented in terms of an integer rating matrix R as follows:

R(i, j) = (r ∈ {1,2,3,4,5}, If a user Ui has given the rating to movie Mj

0, Otherwise

Write a C program to fill in the missing values using the following operations

(a) With random values between 1 to 5.

(b) With row average mapped to the nearest integer

(c) With column average mapped to the nearest integer

(d) With a combination of row and column averages mapped to the nearest integer

Compute and report the deviation of the resulting matrix from the original rating matrix (R) in each case.

CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int **matrix(int m, int n)
{
    int **p = (int **)calloc(m, sizeof(int *));
    for (int i = 0; i < m; i++)
    {
        p[i] = (int *)calloc(n, sizeof(int));
    }
    return p;
}
void display(int **p, int m, int n)
{
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("%d\t", p[i][j]);
        }
        printf("\n");
    }
}
void deviation(int **original, int **second, int m, int n)
{
    int **dev = matrix(m, n);
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            dev[i][j] = original[i][j] - second[i][j];
        }
    }
```

```
    printf("the deviation is:\n");
    display(dev, m, n);
}
int **cond_1(int **p, int m, int n)
{
    int **temp = matrix(m, n);
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            temp[i][j] = p[i][j];
        }
    }

    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (temp[i][j] == 0)
            {
                temp[i][j] = rand() % (5) + 1;
            }
        }
    }
    return temp;
}
int **cond_2(int **p, int m, int n)
{
    int **temp = matrix(m, n);
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            temp[i][j] = p[i][j];
        }
    }
    double s;
    int c;
    for (int i = 0; i < m; i++)
    {
        s = 0.0;
        c = 0;
        for (int j = 0; j < n; j++)
        {

            if (temp[i][j] != 0)
            {
                s = s + temp[i][j];
                c = c + 1;
            }
        }
        int avg = round(s / c);
```

```
        for (int k = 0; k < n; k++)
        {
            if (temp[i][k] == 0)
            {
                temp[i][k] = avg;
            }
        }
    }
    return temp;
}
int **cond_3(int **p, int m, int n)
{
    int **temp = matrix(m, n);
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            temp[i][j] = p[i][j];
        }
    }
    double s;
    int c;
    for (int i = 0; i < n; i++)
    {
        s = 0.0;
        c = 0;
        for (int j = 0; j < m; j++)
        {

            if (temp[j][i] != 0)
            {
                s = s + temp[j][i];
                c = c + 1;
            }
        }
        int avg = round(s / c);
        for (int k = 0; k < m; k++)
        {
            if (temp[k][i] == 0)
            {
                temp[k][i] = avg;
            }
        }
    }
    return temp;
}
int **cond_4(int **p, int m, int n)
{
    int **temp = matrix(m, n);
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
```

```
        {
            temp[i][j] = p[i][j];
        }
    }
    double s1;
    int c1, c2;
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            s1 = 0;
            c1 = 0;
            c2 = 0;
            if (p[i][j] == 0)
            {
                for (int k = 0; k < n; k++)
                {
                    if (p[i][k] != 0)
                    {
                        s1 = s1 + p[i][k];
                        c1 = c1 + 1;
                    }
                }
                for (int l = 0; l < m; l++)
                {
                    if (p[l][j] != 0)
                    {
                        s1 = s1 + p[l][j];
                        c2 = c2 + 1;
                    }
                }
                int avg = round(s1 / (c1 + c2));
                temp[i][j] = avg;
            }
        }
    }
    return temp;
}
int main()
{
    int m, n;
    printf("enter number of users and movies respectively:");
    scanf("%d%d", &m, &n);
    int **original = matrix(m, n);
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            original[i][j] = rand() % (5) + 1;
        }
    }
    printf("original matrix:\n");
```

```
    display(original, m, n);
    int **second = matrix(m, n);
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            second[i][j] = rand() % (6);
        }
    }
    printf("second matrix:\n");
    display(second, m, n);
    int **fill_1 = cond_1(second, m, n);
    printf("matrix after cond1:\n");
    display(fill_1, m, n);
    deviation(original, fill_1, m, n);
    int **fill_2 = cond_2(second, m, n);
    printf("matrix after cond2:\n");
    display(fill_2, m, n);
    deviation(original, fill_2, m, n);
    int **fill_3 = cond_3(second, m, n);
    printf("matrix after cond3:\n");
    display(fill_3, m, n);
    deviation(original, fill_3, m, n);
    int **fill_4 = cond_4(second, m, n);
    printf("matrix after cond4:\n");
    display(fill_4, m, n);
    deviation(original, fill_4, m, n);
}
```