Assignment-1

Vivek Roshan Suggala
M10725902

1. Add the three scores for each student and write the sum in a new column. Discretize the new sum column into five groups using equal width partitioning and assign one of the five grades (A, B, C, D, and F) to students in the five groups (the highest scores get A and the lowest scores get F). Show the grades assigned to each student in a list sorted according to student id. List the counts of each letter grade awarded.

Solution: **Steps followed:**

- The excel file with columns of marks for Physics, Maths, English are to be added in order to get the sum of all the three subjects.
- The bins are to be divided in such a way that, bins are divided with an equal interval with maximum of sum as upper boundary and minimum of sum as lower boundary.
- Get the minimum, maximum values of the sum vector and find the interval value.
- Provide boundaries in such a way that, the bins are divided according to the provided boundaries.
- Discretize the sum vector and give grades according to the bin.
- Count the frequency of each grade and store it in another vector.

Matlab Code:

```
%problem 1
filename= 'C:\Users\suggalvn\Downloads\DataHW1.xlsx'; %path of the file stored in 'filename'
Sid = xlsread(filename,'A:A'); %reading the student-ID's from the excel file to Sid
Sum_of_subjects =
xlsread(filename,strcat('B2:B',num2str(length(Sid)+1)))+xlsread(filename,strcat('C2:C',num2str(l
ength(Sid)+1)))+xlsread(filename,strcat('D2:D',num2str(length(Sid)+1)));
%Adding the three columns; Physics, Maths, English from the excel and
%storing it in Sum_of_subjects
xlswrite(filename,Sum_of_subjects,1,strcat('E2:E',num2str(length(Sid)+1)));
%writing the Sum_of_subjects into the excel file
minimumvalue= min(Sum_of_subjects); %finding out the minimum of Sum among all the students
maximumvalue = max(Sum_of_subjects); %finding out the maximum of Sum among all the students
interval = (maximumvalue-minimumvalue)/5; %calculating the interval with the mentioned formula
edges = [minimumvalue minimumvalue+interval minimumvalue+(interval*2) minimumvalue+(interval*3)
minimumvalue+(interval*4) maximumvalue];
%edges contain the boundaries of the bins to be divided.
%creating the boundaries for the discretize function and storing it in 'edges'
Y = discretize(Sum_of_subjects , edges); %discretizing according to the boundaries(edges)
%the output of Y would be numbers ranging from 1 to 5, as wew gave 6
%boundaries in edges
Y = char(Y); %converting the number vector into character vector
Y( Y==1 )= 'F';Y( Y==2 )= 'D';Y( Y==3 )= 'C';Y( Y==4 )= 'B';Y( Y==5 )= 'A';
%replacing the values accordingly; 1 with F, 2 with D and so on till 5 with A
xlswrite(filename,Y,1,strcat('F2:F',num2str(length(Sid)+1))); %writing the grades to file
Unique_grades = unique(Y); %unique function picks out the uique values of Y and place them in
Unique_grades vector
for a=1:length(Unique_grades) %finding out the frquency of each grade
```

```
   Frquency_of_grades(a)=length(find(Y == Unique_grades(a)));
end
%output:
fprintf('Problem 1 output \n')
output = [string(Sid),Y];
fprintf('Grades according to equal width partitioning are:\n');
disp(output');
fprintf('Counts of each grades, A B C D F respectively are:\n');
disp(Frquency_of_grades)
```

Note: The output vectors are printed as their transposes for the convenience as the transposes occupy less space.

Results obtained:

```
Problem 1 output
Grades according to equal width partitioning are:
  Columns 1 through 11

    "1"    "2"    "3"    "4"    "5"    "6"    "7"    "8"    "9"    "10"    "11"
    "B"    "B"    "D"    "A"    "D"    "C"    "C"    "C"    "A"    "B"     "D"

  Columns 12 through 20

    "12"    "13"    "14"    "15"    "16"    "17"    "18"    "19"    "20"
    "C"     "C"     "A"     "C"     "C"     "D"     "B"     "B"     "C"

  Columns 21 through 29

    "21"    "22"    "23"    "24"    "25"    "26"    "27"    "28"    "29"
    "C"     "D"     "D"     "B"     "C"     "C"     "B"     "C"     "B"

  Columns 30 through 38

    "30"    "31"    "32"    "33"    "34"    "35"    "36"    "37"    "38"
    "C"     "B"     "D"     "A"     "D"     "F"     "D"     "B"     "C"

  Columns 39 through 40

    "39"    "40"
    "A"     "F"

Counts of each grades, A B C D F respectively are:
     5    10    14     9     2
```

Note: The code and the outputs are divided according to the questions for the convenience. The source code for all the problems is written in the same file. Individual codes may not run accordingly because of the loss of continuity.

2. Repeat problem #1 above with the difference that this time use equal frequency partitioning to discretize the sum of points obtained. List the counts of each letter grade awarded. List student ids of those students who would be happier with equal width binning and also of those students' ids that would be happier with equal frequency binning.

Solution: **Steps followed:**

- With the same sum vector obtained by adding the three subjects, we have to divide the vector into 5 bins according to equal frequency binning.
- Equal frequency binning is a method, where there are equal numbers (frequency) of people falling in the each grade.
    - For Example: if we have total 40 values that are to be given grades, according to equal frequency binning, each grade contains 8 students belonging to each bin. There exist few exceptions, where same values fall exactly at the dividing border. Different approaches can be followed at that situation.
- Give grades to the students according to the method followed; find out the frequency of each grade.
- Now, find out the students whose grades had been changed from equal width binning and equal frequency binning and by comparing both the grades obtained, find out the students who are happy with equal width and happy with equal frequency binning separately.

Explanation:

- The code ceil(5 * tiedrank(Sum_of_subjects) / length(Sum_of_subjects)); does the following functionality.
    - tiedrank is a function which gives ranks to the values in a vector. tiedrank(Sum_of_subjects) gives a vector with different ranks assigned to each of the elements of the vector depending upon their values.
    - As I wanted to divide into 5 groups, I multiply the resulting tiedrank vector with 5/length(Sum_of_subjects). By doing so, I can divide the input vector into required bins, keeping in mind that if there are some values repeated exactly at the dividing point, it ceils the value in such a way that all the similar values goes only to one in even though it is equal frequency binning.
    - By this way a same value would not get two different grades, if they lie in the exact bin division point.

Matlab Code:

```
%problem 2
P = ceil(5 * tiedrank(Sum_of_subjects) / length(Sum_of_subjects));
%equal frequency binning using tiedrank
P=char(P);P(P==1)='F'; P(P==2)='D'; P(P==3)='C'; P(P==4)='B'; P(P==5)='A'; %replacing the values
accordingly; 1 with F, 2 with D and so on till 5 with A
xlswrite(filename,P,1,strcat('G2:G',num2str(length(Sid)+1))); %writing the grades to the file
Unique_grades = unique(P);
for i=1:length(Unique_grades)
   frequency2(i)=length(find(P == Unique_grades(i))); %frequency of each grade
end
differences = Y == P; %students whose grades got changed from equal width to equal frequency
for i=1:length(differences) %finding out who are happy for width and who are happy for frequency
```

```
    if(differences(i)==0)
        if(uint8(Y(i)<uint8(P(i))))
            Happier_width_frequency(i)= string('Happy with width');
        else
            Happier_width_frequency(i) = string('Happy with frequency');
        end
    else
        Happier_width_frequency(i) = string('No change');
    end
end
Happier_width_frequency = Happier_width_frequency';
List_happy_equal_width_than_frequency = find(Happier_width_frequency == 'Happy with width');
%listing out the student-ID's who are happy with equal width partitioning
List_happy_equal_frequency_than_width = find(Happier_width_frequency == 'Happy with frequency');
%listing out the student-ID's who are happy with equal frequency partitioning
%output:
fprintf('Problem 2 output \n')
output = [string(Sid),P];
fprintf('Grades according to equal frequency partitioning are:\n');
disp(output');
fprintf('Counts of each grades, A B C D F respectively are:\n');
disp(frequency2);
fprintf('List of people who are happy with equal width partitioning than equal frequency
are:\n');
disp(List_happy_equal_width_than_frequency);
fprintf('List of people who are happy with equal frequency partitioning than equal width
are:\n');
disp(List_happy_equal_frequency_than_width);
```

Note: The output vectors are printed as their transposes for the convenience as the transposes occupy less space

Results obtained:

```
Problem 2 output
Grades according to equal frequency partitioning are:
  Columns 1 through 11

    "1"    "2"    "3"    "4"    "5"    "6"    "7"    "8"    "9"    "10"    "11"
    "A"    "B"    "D"    "A"    "F"    "C"    "D"    "C"    "A"    "B"     "F"

  Columns 12 through 20

    "12"    "13"    "14"    "15"    "16"    "17"    "18"    "19"    "20"
    "C"     "B"     "A"     "D"     "C"     "F"     "B"     "B"     "C"

  Columns 21 through 29

    "21"    "22"    "23"    "24"    "25"    "26"    "27"    "28"    "29"
    "C"     "F"     "D"     "A"     "C"     "C"     "B"     "D"     "B"
  Columns 30 through 38

    "30"    "31"    "32"    "33"    "34"    "35"    "36"    "37"    "38"
    "C"     "A"     "D"     "A"     "F"     "F"     "F"     "A"     "D"

  Columns 39 through 40

    "39"    "40"
    "A"     "F"

Counts of each grades, A B C D F respectively are:
     9     7     9     7     8

List of people who are happy with equal width partitioning than equal frequency are:
     5     7    11    15    17    22    28    34    36    38

List of people who are happy with equal frequency partitioning than equal width are:
     1    13    24    31    37
```

Note: The code and the outputs are divided according to the questions for the convenience. The source code for all the problems is written in the same file. Individual codes may not run accordingly because of the loss of continuity.

3. Compare the grades assigned in problem #1 and #2 above. Make a list of those student ids whose grades changed when the method of binning changed.

Solution: **Steps followed:**

- The aim of this question is to find out the student ID's whose grades have been changed from equal width partitioning to equal frequency partitioning.
- Compare the vectors containing the grades of equal width partitioning and equal frequency partitioning and store the result in another vector.
- According to the code which I have written, the results obtained from comparing both the vectors gives a vector which contain 0's and 1's.
- The grades which are same (not changed) give a 1 and the grades which are changed because of different methods gives a 0.
- From the result vector, find out all the students who have 0 in the comparison result vector.

Matlab Code:

```
%problem 3
output = find(differences == 0); %Students whose grades had been changed, their value would
be 0
%output
fprintf('Problem 3 output \n')
fprintf('Students whose grades had been changed are:\n');
disp(output');
```

Note: The output vectors are printed as their transposes for the convenience as the transposes occupy less space.

Results obtained:

```
Problem 3 output
Students whose grades had been changed are:
  Columns 1 through 13

     1     5     7    11    13    15    17    22    24    28    31    34    36

  Columns 14 through 15

    37    38
```

Note: The code and the outputs are divided according to the questions for the convenience. The source code for all the problems is written in the same file. Individual codes may not run accordingly because of the loss of continuity.

4. Convert the Physics and Maths points to their equivalent z-scores in each column. Sum the three z-scores for each student and use equal frequency binning to create five bins. Assign letter grades to the students and show them in a list sorted by student ids.

Solution: **Steps followed:**

- The excel file with columns of marks for Physics, Maths, English are to be converted to their z-scores and are added in order to get the sum of all the three z-scores for the subjects.
- Divide the z-score sums obtained into five bins according to equal frequency binning, following the exact same method explained in the problem 2.
- Give the letter grades to the students according to the equal frequency partitioning method, each grade to each bin divided.
- Write the output grades according to the z-scores to the excel file accordingly.
- Make a list of student grades sorted according to the student id for the output.

Matlab Code:

```
%problem 4
%Adding the zscores for three columns; Physics, Maths, English from the excel and storing it
in Z_Score_totals
Z_Score_totals =
zscore(xlsread(filename,strcat('B2:B',num2str(length(Sid)+1))))+zscore(xlsread(filename,strc
at('C2:C',num2str(length(Sid)+1))))+zscore(xlsread(filename,strcat('D2:D',num2str(length(Sid
)+1))));
Z_grades = ceil(5 * tiedrank(Z_Score_totals) / length(Z_Score_totals)); %giving grades to
the zscore totals
%replacing the values accordingly; 1 with F, 2 with D and so on till 5 with A
Z_grades=char(Z_grades);Z_grades(Z_grades==1)='F'; Z_grades(Z_grades==2)='D';
Z_grades(Z_grades==3)='C'; Z_grades(Z_grades==4)='B'; Z_grades(Z_grades==5)='A';
xlswrite(filename,Z_grades,1,strcat('H2:H',num2str(length(Sid)+1))); %writing it to excel
file
%output
```

```
fprintf('Problem 4 output \n')
fprintf('Z-scores for the students are:\n');
output = [string(Sid),string(Z_Score_totals)];
disp(output');
output = [string(Sid),Z_grades];
fprintf('Grades according to z-score partitioning are:\n');
disp(output');
```

Note: The output vectors are printed as their transposes for the convenience as the transposes occupy less space.

Results Obtained:

```
Problem 4 output
Z-scores for the students are:
  Columns 1 through 6

    "1"          "2"          "3"            "4"          "5"            "6"
    "0.34885"    "1.0315"     "-1.3825"     "2.3221"     "-1.4943"     "-1.0917"

  Columns 7 through 12

    "7"          "8"          "9"          "10"          "11"          "12"
    "-0.1199"    "0.81556"    "3.8466"     "0.31202"     "-3.2617"     "-1.447"

  Columns 13 through 18

    "13"          "14"         "15"           "16"          "17"          "18"
    "-0.32448"    "2.4819"     "-0.76471"     "-1.4844"     "-1.1762"     "1.9194"

  Columns 19 through 24

    "19"         "20"          "21"           "22"          "23"           "24"
    "1.6763"     "0.61009"     "-0.46523"     "-2.4297"     "-0.85801"     "2.9483"

  Columns 25 through 30

    "25"          "26"          "27"         "28"           "29"         "30"
    "0.058464"    "0.87185"     "1.6761"     "0.011058"     "1.5824"     "0.095563"

  Columns 31 through 36

    "31"         "32"          "33"         "34"         "35"          "36"
    "1.2557"     "-1.1765"     "2.5382"     "-2.981"     "-5.4885"     "-1.9142"

  Columns 37 through 40
```

```
      "37"        "38"       "39"        "40"
      "2.1623"    "-1.84"    "4.9316"    "-3.7958"

    Grades according to z-score partitioning are:
      Columns 1 through 11

        "1"    "2"    "3"    "4"    "5"    "6"    "7"    "8"    "9"    "10"    "11"
        "C"    "B"    "D"    "A"    "F"    "D"    "C"    "B"    "A"    "C"     "F"

      Columns 12 through 20

        "12"    "13"    "14"    "15"    "16"    "17"    "18"    "19"    "20"
        "D"     "C"     "A"     "D"     "D"     "D"     "A"     "B"     "B"

      Columns 21 through 29

        "21"    "22"    "23"    "24"    "25"    "26"    "27"    "28"    "29"
        "C"     "F"     "D"     "A"     "C"     "B"     "B"     "C"     "B"

      Columns 30 through 38

        "30"    "31"    "32"    "33"    "34"    "35"    "36"    "37"    "38"
        "C"     "B"     "D"     "A"     "F"     "F"     "F"     "A"     "F"

      Columns 39 through 40

        "39"    "40"
        "A"     "F"
```

Note: The code and the outputs are divided according to the questions for the convenience. The source code for all the problems is written in the same file. Individual codes may not run accordingly because of the loss of continuity.

5. Compare the grades obtained in #4 and in #2 above. Make a list of students who would be happier with the method in #2 and also a list of those who would be happier with the method in #4. Which method is fairer and why?

Solution: **Steps followed:**

- The aim of this question is to compare the grades that have been changed from equal frequency partitioning to z-score partitioning.
- Compare the vectors containing the grades of z-score partitioning and equal frequency partitioning and store the result in another vector.
- According to the code which I have written, the results obtained from comparing both the vectors gives a vector which contain 0's and 1's.
- Now, compare the grades which have been changed and find out which method is happier for the students (whose grades have been changed).

- List out separately, the students who are happy with the z-scores and the students who are happy with the equal frequency partitioning.

  Explanation:

  - Z-score is fairer than equal frequency in method 2, even though there are many students who are happy with equal frequency and not so happy with the z-scores.
  - Z-score is the method which is measured in standard deviations, and the calculation is done from the 'mean (average)' to the observation (grade or a total).
  - Unlike the equal frequency, z-score metric is a standard metric that calculates the z-score values from the mean and by standard deviations.
  - For ex: let us consider that, physics exam is very tough and most of the people (68%) fall in the range of 40-60 marks with a mean of 50 and at the same time English is easy with 80 marks as mean and 68% people fall in the range of 70-90 in a normal distribution curve.
  - Now, it is not correct to compare both the subjects because the marks and the grades depend on the toughness of the paper but not the totals. They should be graded individually according to the toughness (mean) rather than blindly summing up all the subjects together.
  - Method two takes the sum of all the subjects, hence I think it is not fair to take the sum for the grading purpose.
  - Hence it is wise to give the grades to the subjects individually (like the process in z-score) , rather than giving grades to their sum.

Matlab Code:

```
%problem 5
differences_4_2 = Z_grades == P; %students whose grades got changed from equal frequency and
z-score grades
for i=1:length(differences_4_2) %finding out who are happy for zscore and who are happy for
frequency
   if(differences_4_2(i)==0)
       if(uint8(Z_grades(i))<uint8(P(i)))
           Happier_frequency_zscore(i)= string('Happy with Z-scores');
       else
           Happier_frequency_zscore(i) = string('Happy with frequency');
       end
   else
       Happier_frequency_zscore(i) = string('No change');
   end
end
Happier_frequency_zscore = Happier_frequency_zscore';
%listing out the student-ID's who are happy with equal frequency partitioning
List_happy_equal_frequency_than_zscore = find(Happier_frequency_zscore == 'Happy with
frequency');
%listing out the student-ID's who are happy with zscore partitioning
List_happy_equal_Z_score_than_frequency = find(Happier_frequency_zscore == 'Happy with Z-
scores');
%output:
fprintf('Problem 5 output \n');
fprintf('List of people who are happy with equal frequency partitioning than Z-score
are:\n');
disp(List_happy_equal_frequency_than_zscore);
fprintf('List of people who are happy with z-score partitioning rather than equal frequency
partitioning are:\n');
disp(List_happy_equal_Z_score_than_frequency);
```

Note: The output vectors are printed as their transposes for the convenience as the transposes occupy less space.

Results Obtained:

```
Problem 5 output
List of people who are happy with equal frequency partitioning than Z-score are:
     1
     6
    10
    12
    13
    16
    31
    38
List of people who are happy with z-score partitioning rather than equal frequency
partitioning are:
     7
     8
    17
    18
    20
    26
    28
```

Note: The code and the outputs are divided according to the questions for the convenience. The source code for all the problems is written in the same file. Individual codes may not run accordingly because of the loss of continuity.

6. Consider a student who is happier with the method in #4 compared to the method in #2. Briefly explain why his being happier is justified. Also consider a student who is happier with the method in #2 and briefly explain why he/she is or is not justified.

Solution: **Explanation:**

- Z-score is fairer than equal frequency partitioning in method 2, even though there are many students who are happy with equal frequency and not so happy with the z-scores.
- Z-score is the method which is measured in standard deviations, and the calculation is done from the 'mean (average)' to the observation (grade or a total).
- Unlike the equal frequency, z-score metric is a standard metric that calculates the z-score values from the mean and by standard deviations.

- Let us consider an example of finding out the distance of two points; consider the marks of a student as the coordinates in a 3D-space.
- Now, when finding out the distance between two points, it is not advisable to compare two unequally tough subjects' marks. One subject may be tough and tend to have very low values and the other might be easy and may have higher values.
- Z-score eliminates the problem of unequal comparability because it computes the z-scores according to the standard deviations from the mean.
- It is wiser to grade individual subject using the z-scores. In our given data in excel, student 1 got an A in the equal frequency partitioning and a C in z-scores because z-scores are calculated from the mean using standard deviations.
- Hence the students who are happier with method 4 are being justified because the measure is the standard deviations from the individual means, rather than depending upon the totals (which is done in method #2).
- These students scored are according to the toughness and are above the mean level in every subject.
- Consider an example; a student scored 25 in one subject and 100 in another. He adds to 125 and another student scores 55 in one and 55 in another. Let's consider mean to be 50 in each of the subjects.
- Now with equal frequency first student gets an A and the another might not get an A. But when individual grades are seen, the first student fails in a subject with an 'F'
- Hence, a student with z-score happier is justified and the student who is happy with the method 2 is not justified because of the reason specified above.

Note: The code and the outputs are divided according to the questions for the convenience. The source code for all the problems is written in the same file. Individual codes may not run accordingly because of the loss of continuity.

7. Select two students for each type of grade from the list of grades assigned in #1 above. Create a 10X10 table of all pairwise distances using (i) Euclidean distance.

Solution: **Steps followed:**

- Create a matrix with student ID's and their grades (from equal width partitioning).
- Now, we have to retrieve 10 students from the list of 40, in such a way that we retrieve two students out of each grade.
- It can be done with the help of a for-loop. We already have an array with all the possible grades (A, B, C, D, and E). Hence initialize a counter variable to 1, take one grade from the array and add the entire row to a new array.
- Continue this process till the counter is 2. Whenever it reaches two, take the next grade for the comparison.
- In this way continue for all the grades, two students for each grade forming 10 students.
- Now, we have to bring all the three marks of those 10 students in a 10*3 vector. 3 be the three marks and 10 rows for 10 students.
- It can be done using a simple for loop, comparing the student list with the excel data and the list of 10 students which is obtained in the step above.
- Now Euclidean distance can be calculated easily with the help of 'pdist(Q(:,2:4))' where Q is the 10*3 matrix with 10 rows of students with 3 rows of their individual marks.
- In order to make it to a square form, that is to compare one student with the other nine and so on, squareform function can be used.
- Hence 'euclidean = squareform(pdist(Q(:,2:4)));' this piece of code fulfills our requirement creating a 10*10 matrix.

Matlab Code:

```
%problem 7
Matrix_Sid_grades = [string(Sid),Y]; %Matrix_Sid_grades contains Sid and grades(according to
equal width partitioning)
j=1;
for g=1:5 %retrieving two students of each grade and storing in Desired_Matrix
    counter=1;
    for i=1:length(Sid)
        if(Matrix_Sid_grades(i,2)==Unique_grades(g) && counter<=2)
            Desired_Matrix(j) = str2double(Matrix_Sid_grades(i,1));
            j=j+1;
            Matrix_Sid_grades(i,2)=' ';
            counter= counter+1;
        end
    end
end
Desired_Matrix = Desired_Matrix'; %making the transpose of Desired_Matrix, from 1*40 to 40*1
X = xlsread(filename,'A2:D41'); %reading the first four columns from excel to 'X'
for j=1:10 %creating a matrix Q, such that it contains only 3 columns of
%marks(Physics,Maths,English) for the
    %students who got selected in the Desired_Matrix
    for i=1:length(Sid)
        if (X(i,1)== Desired_Matrix(j))
            Q(j,:) =  X(i,:);
        end
    end
end
euclidean = squareform(pdist(Q(:,2:4))); %calculating euclidean distance and forimg a 10*10
matrix
mahalanobi = squareform(pdist(Q(:,2:4),'mahalanobis')); %calculating mahalanobi distance
forimg a 10*10 matrix
euclidean_distance = euclidean;
mahalanobi_distance = mahalanobi;
%output:
fprintf('Problem 7 output \n');
fprintf('Euclidean distance 10*10 matrix: \n');
disp(euclidean);
%fprintf('Mahalanobi distance 10*10 matrix: \n');
%disp(mahalanobi);
```

Note: The output vectors are printed as their transposes for the convenience as the transposes occupy less space.

Results Obtained:

```
Problem 7 output
Euclidean distance 10*10 matrix:
  Columns 1 through 7
```

```
         0   31.3688   15.0000   39.1152   35.6371   71.5821   56.6392
   31.3688         0   39.8623   26.0384   45.8912   60.9590   62.1611
   15.0000   39.8623         0   37.7492   24.5967   66.8506   47.4658
   39.1152   26.0384   37.7492         0   28.6356   38.3406   41.1096
   35.6371   45.8912   24.5967   28.6356         0   45.6289   24.7790
   71.5821   60.9590   66.8506   38.3406   45.6289         0   33.1662
   56.6392   62.1611   47.4658   41.1096   24.7790   33.1662         0
   74.7262   64.8074   68.4763   40.2244   46.1086   10.7703   34.9857
  104.0817  104.1393   93.7443   79.1265   69.4046   48.8774   48.5489
  100.8464   96.7781   92.2226   72.2496   67.8233   37.7094   46.8402

  Columns 8 through 10

   74.7262  104.0817  100.8464
   64.8074  104.1393   96.7781
   68.4763   93.7443   92.2226
   40.2244   79.1265   72.2496
   46.1086   69.4046   67.8233
   10.7703   48.8774   37.7094
   34.9857   48.5489   46.8402
         0   44.5533   34.4384
   44.5533         0   15.3948
   34.4384   15.3948         0
```

Note: The code and the outputs are divided according to the questions for the convenience. The source code for all the problems is written in the same file. Individual codes may not run accordingly because of the loss of continuity.

8. Select the four student pairs with the minimum distance values from the Euclidean distance.

Solution: **Steps followed:**

- Find out the minimum value in the Euclidean 10*10 matrix and take its indexes, with the indexes find out the student ID's from the 10 student ID's list.
- Make the minimum value as Infinity and also make its symmetric value, that is; if minimum is found at indexes 1*3, as the matrix is a symmetric matrix 3*1 also has the same value.
- So convert the least values to infinity, find out the next minimum, save its indexes to find the student ID's and make the two values infinite.
- Continue this process till you get the four least minimums with the pair associated with those minimum values.

Matlab Code:

```
%problem 8
euclidean_distance(euclidean_distance == 0 ) = Inf; %making the 0's to infinity
mahalanobi_distance(mahalanobi_distance == 0 ) = Inf; %making the 0's to infinity
```

```
for i=1:4
    %making certain manipulations such as retrieving top 4 minimum values
    %and finding out the student pair from euclidean_distance and mahalanobi_distance
    [M,I]= min(euclidean_distance);
    [M,J]= min(min(euclidean_distance));
    euclidean_distance(I,J)= Inf;
    euclidean_distance(J,I)= Inf;
    I = I(J);
    euclidean_pairs(i,1) = M;
    euclidean_pairs(i,2) = Desired_Matrix(I);
    euclidean_pairs(i,3) = Desired_Matrix(J);
    %euclidean_pair consists of student pairs for top 4 minimum euclidean distances

    [M,I]= min(mahalanobi_distance);
    [M,J]= min(min(mahalanobi_distance));
    mahalanobi_distance(I,J)= Inf;
    mahalanobi_distance(J,I)= Inf;
    I = I(J);
    mahalanobi_pair(i,1) = M;
    mahalanobi_pair(i,2) = Desired_Matrix(I);
    mahalanobi_pair(i,3) = Desired_Matrix(J);
    %mahalanobi_pair consists of student pairs for top 4 minimum mahalanobi distances
end
%output:
fprintf('Problem 8 output \n');
fprintf('four student pairs with minimum Euclidean distance matrix: \n');
output = [string(double(euclidean_pairs(:,1))) ,string(int16(euclidean_pairs(:,2))),
string(int16(euclidean_pairs(:,3))) ];
disp(output);
%fprintf('four student pairs with minimum mahalanobi distance matrix: \n');
%output = [string(double(mahalanobi_pair(:,1))) ,string(int16(mahalanobi_pair(:,2))),
%string(int16(mahalanobi_pair(:,3))) ];
%disp(output);
```

Note: The output vectors are printed as their transposes for the convenience as the transposes occupy less space.

Results Obtained:

```
Problem 8 output
four student pairs with minimum Euclidean distance matrix:
    "10.7703"    "5"     "7"
    "15"         "1"     "4"
    "15.3948"    "40"    "35"
    "24.5967"    "6"     "1"
```