AFRAAZ HUSSAIN
20BDS0374
DESIGN AND ANALYSIS OF ALGORITHMS
LAB ASSESSMENT 1

QUESTION:

Design and implement an algorithm using brute force approach that finds the top and the least scores of students from an online Quiz. Take scores as input and store in an array.

PSEUDOCODE:

```
studentScore = []
input(studentScore)
assign lowest = highest = studentScore[0]

for score in studentScore:
    if score > highest:
        set it as new highest
    if score < lowest
        set it as new lowest

print(lowest, highest)
```

CODE:

```
#include <stdio.h>



struct studentDetails
{
    int studentID, studentScore;
};

void main()
{
    printf("Name: Afraaz Hussain\nAdmission number: 20BDS0374\n\n\n");

    int studentCount, highestScore, lowestScore, highestScoreStudent,
lowestScoreStudent;

    printf("Enter the total number of students: ");
    scanf("%d", &studentCount);
    printf("\n");
    struct studentDetails studentDetail[studentCount];

    for (int index = 0; index < studentCount; index++)
    {
        studentDetail[index].studentID = index + 1;
        printf("Enter the score of student %d: ", index + 1);
```

```
        scanf("%d", &studentDetail[index].studentScore);
    }

    highestScore = studentDetail[0].studentScore;
    highestScoreStudent = studentDetail[0].studentID;
    lowestScore = studentDetail[0].studentScore;
    lowestScoreStudent = studentDetail[0].studentID;
    for (int index = 0; index < studentCount; index++)
    {
        if (studentDetail[index].studentScore > highestScore)
        {
            highestScore = studentDetail[index].studentScore;
            highestScoreStudent = studentDetail[index].studentID;
        }
        if (studentDetail[index].studentScore < lowestScore)
        {
            lowestScore = studentDetail[index].studentScore;
            lowestScoreStudent = studentDetail[index].studentID;
        }
    }

    printf("\nThe highest score was %d by student %d", highestScore,
highestScoreStudent);
    printf("\nThe lowest score was %d by student %d", lowestScore,
lowestScoreStudent);
}
```

OUTPUT:

```
Name: Afraaz Hussain
Admission number: 20BDS0374


Enter the total number of students: 5

Enter the score of student 1: 10
Enter the score of student 2: 23
Enter the score of student 3: 97
Enter the score of student 4: 03
Enter the score of student 5: 74

The highest score was 97 by student 3
The lowest score was 3 by student 4
```

AFRAAZ HUSSAIN
20BDS0374
DESIGN AND ANALYSIS OF ALGORITHMS
LAB ASSESSMENT 1

QUESTION:

Take ‘n’ historic sites in Tamil Nadu as input. Design and implement an algorithm using brute force approach that identifies the shortest possible route for a traveler to visit these sites and come back to his starting point.

PSEUDOCODE:

```
findDistance(xOne, xTwo, yOne, yTwo):
    //Find distance using distance formula
    return distance

main():
    cityCount.input()
    create variables to hold leastDistance and rout

    for row from 1 to cityCount - 1:
        for column from 0 to cityCount - 1:
            currentDistance = 0
            temporary = cityList[column]
            cityList[column] = cityList[column + 1]
            cityList[column + 1] = temporary

            //Find the distance of all the elements in this list by using the
distance function
            if leastDistance > currentDistance:
                leastDistance = currentDistance
                route = currentRoute
```

CODE:

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



float calculateDistance(float xOne, float yOne, float xTwo, float yTwo)
{
    float distance = ((xTwo - xOne) * (xTwo - xOne)) + ((yTwo - yOne) * (yTwo
- yOne));
    return sqrt(distance);
}



void main()
```

AFRAAZ HUSSAIN
20BDS0374
DESIGN AND ANALYSIS OF ALGORITHMS
LAB ASSESSMENT 1

```
{
    printf("Name: Afraaz Hussain\nAdmission number: 20BDS0374\n\n\n");

    int numberOfCities, row, column, index, origin;
    printf("Please enter the total number of cities: ");
    scanf("%d", &numberOfCities);
    printf("\n");

    float cityLocation[numberOfCities][2],
cityDistance[numberOfCities][numberOfCities];
    for (index = 0; index < numberOfCities; index++)
    {
        printf("Please enter the location of city %d: ", index + 1);
        scanf("%f %f", &cityLocation[index][0], &cityLocation[index][1]);
    }
    for (row = 0; row < numberOfCities; row++)
    {
        for (column = 0; column < numberOfCities; column++)
        {
            cityDistance[row][column] =
calculateDistance(cityLocation[row][0], cityLocation[row][1],
cityLocation[column][0], cityLocation[column][1]);
        }
    }

    printf("\nEnter the city you would like to start from: ");
    scanf("%d", &origin);
    origin--;
    int numberCombinations[numberOfCities - 1], counter = 0;
    for (index = 0; index < numberOfCities; index++)
    {
        if (origin != index)
        {
            numberCombinations[counter] = index;
            counter++;
        }
    }

    float leastDistance = -1.0, currentLeastDistance;
    int temporary, bestRoute[numberOfCities + 1];

    for (row = 1; row <= numberOfCities - 1; row++)
    {
        for (column = 0; column < numberOfCities - 2; column++)
        {
            currentLeastDistance = 0;
```

```
            temporary = numberCombinations[column];
            numberCombinations[column] = numberCombinations[column + 1];
            numberCombinations[column + 1] = temporary;

            currentLeastDistance = cityDistance[origin][numberCombinations[0]]
+ cityDistance[origin][numberCombinations[numberOfCities - 2]];
            for (index = 0; index < numberOfCities - 2; index++)
currentLeastDistance +=
cityDistance[numberCombinations[index]][numberCombinations[index + 1]];
            if (leastDistance == -1.0)
            {
                leastDistance = currentLeastDistance;
                bestRoute[0] = origin;
                bestRoute[numberOfCities] = origin;
                for (index = 1; index < numberOfCities; index++)
bestRoute[index] = numberCombinations[index - 1];
            }
            else if (leastDistance > currentLeastDistance)
            {
                leastDistance = currentLeastDistance;
                bestRoute[0] = origin;
                bestRoute[numberOfCities] = origin;
                for (index = 1; index < numberOfCities; index++)
bestRoute[index] = numberCombinations[index - 1];
            }
        }
    }

    printf("\nthe least distance was calculated to be: %f\n", leastDistance);
    printf("To achieve this, you'll have to use the following route: %d",
origin + 1);
    for (index = 1; index < numberOfCities + 1; index++) printf(" -> %d",
bestRoute[index] + 1);
}
```

AFRAAZ HUSSAIN
20BDS0374
DESIGN AND ANALYSIS OF ALGORITHMS
LAB ASSESSMENT 1

OUTPUT:

```
Name: Afraaz Hussain
Admission number: 20BDS0374


Please enter the total number of cities: 5

Please enter the location of city 1: 2 0
Please enter the location of city 2: 0 0
Please enter the location of city 3: 0 3
Please enter the location of city 4: 3 7
Please enter the location of city 5: 7 4

Enter the city you would like to start from: 3

the least distance was calculated to be: 21.403124
To achieve this, you'll have to use the following route: 3 -> 4 -> 5 -> 1 -> 2 -> 3
```

AFRAAZ HUSSAIN
20BDS0374
DESIGN AND ANALYSIS OF ALGORITHMS
LAB ASSESSMENT 1

QUESTION:

Design and implement an algorithm using divide and conquer approach that finds the maximum and minimum element from the elements stored in an array.

PSEUDOCODE:

```
find minimumAndMaximum(numberArray, lowest, hhighest, minimum, maximum):

    //Condition when there's only one lement
    if lowest and highest are same:
        set minimum = maximum = numberArray[lowest]
        return

    //Condition when there are only two elements
    if lowest + 1 is highest:
        find minimum and maximum
        return

    //Condition when there are more than two elements
    middleElement = lowest + highest / 2
    findMinimumAndMaximum(numberArray, lowest, middleElement, min, max)
    findMinimumAndMaximum(numberArray, middleElement + 1, highest, min, max)
    if (numberArray[middleElement] > max) max = numberArray[middleElement];
    if (numberArray[middleElement] < min) min = numberArray[middleElement];

for row in range:
    for column in range:
        print(element in [row][column])
    print(newLine)
```

CODE:

```
#include <stdio.h>



void findMinimumAndMaximum(int numberArray[], int low, int high, int *min, int
*max)
{
    if (low == high)
    {
        *min = numberArray[low];
        *max = numberArray[low];
        return;
    }
```

```
    if (low + 1 == high)
    {
        if (numberArray[low] > numberArray[high])
        {
            *min = numberArray[high];
            *max = numberArray[low];
        }

        else
        {
            *min = numberArray[low];
            *max = numberArray[high];
        }
        return;
    }

    int middleElement = (low + high) / 2;
    findMinimumAndMaximum(numberArray, low, middleElement, min, max);
    findMinimumAndMaximum(numberArray, middleElement + 1, high, min, max);

    if (numberArray[middleElement] > *max) *max = numberArray[middleElement];
    if (numberArray[middleElement] < *min) *min = numberArray[middleElement];
}



void main()
{
    printf("Name: Afraaz Hussain\nAdmission number: 20BDS0374\n\n\n");

    int numberOfElements, min, max;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &numberOfElements);

    int numberArray[numberOfElements];

    printf("Enter all the elements in the array: ");
    for (int index = 0; index < numberOfElements; index++) scanf("%d",
&numberArray[index]);

    findMinimumAndMaximum(numberArray, 0, numberOfElements - 1, &min, &max);
    printf("\n\nSmallest element in the given array: %d", min);
    printf("\nLargest element in the given array: %d", max);
}
```

AFRAAZ HUSSAIN
20BDS0374
DESIGN AND ANALYSIS OF ALGORITHMS
LAB ASSESSMENT 1

OUTPUT:

```
Name: Afraaz Hussain
Admission number: 20BDS0374


Enter the number of elements in the array: 6
Enter all the elements in the array: 2 0 0 3 7 4


Smallest element in the given array: 0
Largest element in the given array: 7
```

AFRAAZ HUSSAIN
20BDS0374
DESIGN AND ANALYSIS OF ALGORITHMS
LAB ASSESSMENT 1

QUESTION:

Write a menu driven program to implement matrix multiplication and Strassen's matrix multiplication algorithm using divide and conquer approach.

PSEUDOCODE:

- Divide and conquer matrix multiplication:

```
divideAndConquer(matrixOne, matrixTwo, matrixThree, order):
    for row in range order:
        for column in range order:
            matrizThree[row][column] = 0
            for dimension in range order:
                matrixThree[row][column] += matrixOne[row][dimension] *
matrixTwo[dimension][column]
    for row in range order:
        for column in range order:
            print(matrixThree[row][column])
```

- Strassen's matrix multiplication:

```
strassenMatrixMultiplication(int A, int B, int C, int order)

if order == 1 then
    C = C + (A) * (B)
else
    strassenMatrixMultiplication(A, B, C, order / 4)
    strassenMatrixMultiplication(A, B + (order / 4), C + (order / 4),
order / 4)
    strassenMatrixMultiplication(A + 2 * (order / 4), B, C + 2 * (order
/ 4), order / 4)
    strassenMatrixMultiplication(A + 2 * (order / 4), B + (order / 4),
C + 3 * (order / 4), order / 4)
    strassenMatrixMultiplication(A + (order / 4), B + 2 * (order / 4),
C, order / 4)
    strassenMatrixMultiplication(A + (order / 4), B + 3 * (order / 4),
C + (order / 4), order / 4)
    strassenMatrixMultiplication(A + 3 * (order / 4), B + 2 * (order /
4), C + 2 * (order / 4), order / 4)
    strassenMatrixMultiplication(A + 3 * (order / 4), B + 3 * (order /
4), C + 3 * (order / 4), order / 4)
```

CODE:

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

AFRAAZ HUSSAIN
20BDS0374
DESIGN AND ANALYSIS OF ALGORITHMS
LAB ASSESSMENT 1

```
boolean isPowerOfTwo(int number)
{
    if (number == 0) return 0;
    while (number != 1)
    {
        if (number % 2 != 0) return 0;
        number = number / 2;
    }
    return 1;
}

int assignedOrder;
void divideAndConquer(int matrixOne[assignedOrder][assignedOrder], int
matrixTwo[assignedOrder][assignedOrder], int
resultantMatrix[assignedOrder][assignedOrder], int order)
{
    for (int row = 0; row < order; row++)
    {
        for (int column = 0; column < order; column++)
        {
            resultantMatrix[row][column] = 0;
            for (int dimension = 0; dimension < order; dimension++)
            {
                resultantMatrix[row][column] += (matrixOne[row][dimension] *
matrixTwo[dimension][column]);
            }
        }
    }

    printf("\n\nThe product matrix is...\n");
    for (int row = 0; row < order; row++)
    {
        for (int column = 0; column < order; column++) printf("%d ",
resultantMatrix[row][column]);
        printf("\n");
    }
}

void matrixMultiplication()
{
    printf("\e[1;1H\e[2J");
    printf("Name: Afraaz Hussain\nAdmission number: 20BDS0374\n\n\n");
```

```
    int order;
    printf("Enter the order of the square matrices: ");
    scanf("%d", &order);
    int matrixOne[order][order], matrixTwo[order][order],
resultantMatrix[order][order], row, column;
    printf("\n\nEnter the elements of the first matrix...\n");
    for(row = 0; row < order; row++)
    {
        for(column = 0; column < order; column++) scanf("%d",
&matrixOne[row][column]);
    }
    printf("\n\nEnter the elements of the second matrix...\n");
    for(row = 0; row < order; row++)
    {
        for(column = 0; column < order; column++) scanf("%d",
&matrixTwo[row][column]);
    }

    assignedOrder = order;
    divideAndConquer(matrixOne, matrixTwo, resultantMatrix, order);
}

void strassenMatrixMultiplication()
{
    printf("\e[1;1H\e[2J");
    printf("Name: Afraaz Hussain\nAdmission number: 20BDS0374\n\n\n");

    int order;
    printf("Enter the order of the square matrices: ");
    scanf("%d", &order);
    if (isPowerOfTwo(order) && order == 2)
    {
        int matrixOne[order][order], matrixTwo[order][order],
resultantMatrix[order][order], row, column;
        printf("\n\nEnter the elements of the first matrix...\n");
        for(row = 0; row < order; row++)
        {
            for(column = 0; column < order; column++) scanf("%d",
&matrixOne[row][column]);
        }
        printf("\n\nEnter the elements of the second matrix...\n");
        for(row = 0; row < order; row++)
        {
            for(column = 0; column < order; column++) scanf("%d",
&matrixTwo[row][column]);
        }
```

```
        int elements[7];
        elements[0] = (matrixOne[0][0] + matrixOne[1][1]) * (matrixTwo[0][0] +
matrixTwo[1][1]);
        elements[1] = (matrixOne[1][0] + matrixOne[1][1]) * matrixTwo[0][0];
        elements[2] = matrixOne[0][0] * (matrixTwo[0][1] + matrixTwo[1][1]);
        elements[3] = matrixOne[1][1] * (matrixTwo[1][0] + matrixTwo[0][0]);
        elements[4] = (matrixOne[0][0] + matrixOne[0][1]) * matrixTwo[1][1];
        elements[5] = (matrixOne[1][0] - matrixOne[0][0]) * (matrixTwo[0][0] +
matrixTwo[0][1]);
        elements[6] = (matrixOne[0][1] - matrixOne[1][1]) * (matrixTwo[1][0] +
matrixTwo[1][1]);

        resultantMatrix[0][0] = elements[0] + elements[3] - elements[4] +
elements[6];
        resultantMatrix[0][1] = elements[2] + elements[4];
        resultantMatrix[1][0] = elements[1] + elements[3];
        resultantMatrix[1][1] = elements[0] + elements[1] - elements[2] +
elements[5];

        printf("\n\nThe product matrix is...\n");
        for(row = 0; row < order; row++)
        {
            for(column = 0; column < order; column++)
            {
                printf("%d ", resultantMatrix[row][column]);
            }
            printf("\n");
        }
    }
    else
    {
        printf("\n\nOrder of your matrices must be a power of 2");
        sleep(3);
        strassenMatrixMultiplication();
    }
}

void menu()
{
    printf("\e[1;1H\e[2J");
    printf("Name: Afraaz Hussain\nAdmission number: 20BDS0374\n\n\n");

    while (1)
    {
        printf("Please sleect an option...\n");
```

```
        printf("(A) Multiplication using divide and conquer approach\n");
        printf("(B) Strassen's matrix multiplication\n\n");
        printf("Your choice: ");
        char option, runAgain;
        scanf(" %c", &option);

        switch (option)
        {
        case 'A':
            matrixMultiplication();
            break;

        case 'B':
            strassenMatrixMultiplication();
            break;

        default:
            printf("\nThat was not a valid option.");
            sleep(3);
            menu();
            break;
        }

        printf("\n\n\nWould you like to run the program again? [Y / N]: ");
        scanf(" %c", &runAgain);
        if (runAgain == 'Y' || runAgain == 'y') menu();
        else
        {
            printf("\nExiting the program...");
            sleep(2);
            break;
        }
    }


}



void main()
{
    printf("\e[1;1H\e[2J");
    printf("Name: Afraaz Hussain\nAdmission number: 20BDS0374\n\n\n");

    menu();
}
```

OUTPUT:

- Menu:

```
Name: Afraaz Hussain
Admission number: 20BDS0374


Please sleect an option...
(A) Multiplication using divide and conquer approach
(B) Strassen's matrix multiplication

Your choice: C

That was not a valid option.
```

- Divide and conquer matrix multiplication:

```
Name: Afraaz Hussain
Admission number: 20BDS0374


Enter the order of the square matrices: 2


Enter the elements of the first matrix...
1 2
2 3


Enter the elements of the second matrix...
3 4
4 5


The product matrix is...
11 14
18 23
```

- Strassen's matrix multiplication:

```
Name: Afraaz Hussain
Admission number: 20BDS0374


Enter the order of the square matrices: 2


Enter the elements of the first matrix...
1 2
3 4


Enter the elements of the second matrix...
5 6
7 8


The product matrix is...
59 38
83 108
```