National University of Computer & Emerging Sciences, Karachi
Fall-2023 FAST School of Computing
Midterm-II
6th November 2023, 11:30 am – 12:30 pm

Course Code:CS1002	**Course Name:** Programming Fundamentals
Instructor Name: Mr. M. Shahzad, Dr. Farooque, Dr. Abdul Aziz, Mr. Zain, Mr. Basit, Ms. Sobia, Mr. Farooq Zaidi, Mr. M. Kariz	
Student Roll No:	**Section:**

Instructions:

- Return the question paper and make sure to keep it inside your answer sheet.
- Read each question completely before answering it. There are **three questions and two pages (front plus back)**.
- In case of any ambiguity, you may make assumptions. However, your assumption should not contradict any statement in the question paper.
- Do not write anything on the question paper (except your ID and group).

Total Time: 1 Hour **Max Points**: 60

Question#1: **[16 points, CLO2, 15 mins]**

Consider the function below which receives a 2D "5x5" matrix as argument:

```
#define N 5
void function(int mat[N][N]) {
      int i,j;
    for (i = 0; i < N; i++) {
        for (j = 1; j < N; j++) {
            int temp = mat[i][j];
            mat[i][j] = mat[i][j+1];
            mat[i][j+1] = temp;
        }
    }
}
```

The function is expected to give the desired result based on the test case below.

Expected Argument/ input:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Expected Work from Function/ output:

1	6	11	16	21
2	7	12	17	22
3	8	13	18	23
4	9	14	19	24
5	10	15	20	25

You need to determine if the provided function will give the expected output. If not, then provide corrected code that accepts input data and generates the expected output as given below.

---Turn the page for Q2 & Q3---

```
void function(int mat[N][N]) {
   int i, j;
   for (i = 0; i < N; i++) {
      for (j = i+1; j < N; j++) {
         int temp = mat[i][j];
         mat[i][j] = mat[j][i];
         mat[j][i] = temp;
      }
   }
}
```

Rubrics
1 – Identifying as NO (5 marks)
2 – Modifying the loop (5 marks)
3 – Modifying the swap statements (5 marks)

Question#2: **[20 points, CLO1, 20 mins]**

Askari Software House is recruiting new employees. All 50 candidates must go through several tests and interviews to be eligible for the job position. You are given data in the following format, and it is assumed that these variables have data assigned already.

int data[50][4]; *//This stores 50 candidates information, i.e. ID, Analytical test score, Problem solving test score, Interview score*
float GPA[50];
int salaryDemand[50];

Write a C language function prototype and definition that accepts given data and prints statistics.

a) Total number of candidates with GPA higher than 3.0 and more than 50 score in analytical test.
b) The percentage of candidates that have cumulative test score higher than **X** and salary lower than **Y.** (X and Y are user entered).
c) Average salary demand for the post throughout the candidates,
d) Median is a value that is the middle value in a sorted set. You are required to find and print median GPA from all the candidates' GPA.

Rubrics:

5 points for correct traverse and print of number of candidates with higher than 3.0 GPA and analytical score more than 50
5 points for percentage of of candidates with cumulative test score higher than X and salary lower than Y
5 points for average salary demand calculation
5 points for Sorting the array and printing the value on 25 or 24th position.

Solution:

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

void printStatistics(int data[50][4], float GPA[50], int salaryDemand[50], int X, int Y) {
    // a) Total number of candidates with GPA higher than 3.0 and more than 50 score in analytical test.
    int totalHighGPA = 0;
    // b) The percentage of candidates that have cumulative test score higher than X and salary lower than Y.
    int totalQualified = 0;
    // c) Average salary demand for the post throughout the candidates.
    double totalSalary = 0;

    // a) & b) Counting candidates and qualified candidates
    for (int i = 0; i < 50; i++) {
        if (GPA[i] > 3.0 && data[i][1] > 50) {
            totalHighGPA++;
        }
        if ((data[i][1] + data[i][2] + data[i][3]) > X && salaryDemand[i] < Y) {
            totalQualified++;
        }
        // c) Calculating total salary
        totalSalary += salaryDemand[i];
    }

    // c) Calculate average salary
    double averageSalary = totalSalary / 50;

            // d) Median GPA from all the candidates' GPA.
    float sortedGPA[50];
    // Copy GPA array for sorting
    for (int i = 0; i < 50; i++) {
        sortedGPA[i] = GPA[i];
    }
    // Sorting GPA array in ascending order
    for (int i = 0; i < 50; i++) {
        for (int j = i + 1; j < 50; j++) {
            if (sortedGPA[i] > sortedGPA[j]) {
                float temp = sortedGPA[i];
                sortedGPA[i] = sortedGPA[j];
                sortedGPA[j] = temp;
            }
        }
    }

    // d) Find and print median GPA
    float medianGPA;
            medianGPA = sortedGPA[50 / 2];


    // Print statistics
    printf("a) Total number of candidates with GPA higher than 3.0 and more than 50 score in analytical test:
%d\n", totalHighGPA);
```

```
    printf("b) Percentage of candidates that have cumulative test score higher than %d and salary lower than %d: %.2f%%\n", X, Y, (totalQualified / 50.0) * 100);
    printf("c) Average salary demand for the post throughout the candidates: %.2lf\n", averageSalary);
    printf("d) Median GPA from all the candidates' GPA: %.2f\n", medianGPA);
}

int main() {
    // Example usage
    int data[50][4] = {{57, 12, 35, 8}, {63, 46, 17, 31}, {69, 42, 5, 28}, {64, 77, 19, 50}, {73, 26, 41, 16}, {89, 11, 33, 7}, {72, 45, 18, 30}, {70, 43, 4, 27}, {65, 38, 20, 48}, {2, 25, 40, 15}, {48, 13, 36, 9}, {24, 47, 16, 32}, {8, 41, 6, 29}, {13, 36, 21, 49}, {4, 27, 42, 17}, {50, 12, 34, 7}, {53, 46, 19, 31}, {11, 44, 5, 28}, {16, 39, 22, 48}, {3, 26, 40, 14}, {57, 10, 32, 6}, {25, 48, 15, 33}, {7, 40, 8, 30}, {14, 37, 20, 49}, {51, 24, 39, 13}, {49, 11, 33, 9}, {26, 49, 17, 34}, {9, 42, 7, 29}, {17, 40, 23, 50}, {5, 28, 43, 18}, {31, 15, 37, 10}, {12, 35, 21, 46}, {6, 29, 44, 19}, {33, 16, 38, 11}, {19, 32, 8, 41} };
    float GPA[50] = {3.3, 0.8, 1.2, 3.5, 2.1, 0.6, 1.8, 2.3, 1.0, 1.2, 3.1, 2.3, 3.7, 3.9, 3.5, 0.2, 3.6, 1.8, 0.6, 3.1, 1.2, 2.3, 3.5, 0.8, 1.2, 2.3, 3.8, 3.5, 2.1, 0.6, 2.0, 3.1, 2.3, 0.8, 3.5, 1.2, 0.6, 2.1, 1.9, 2.3, 1.2, 3.6, 3.5, 1.8, 2.3, 1.0, 3.1, 2.1, 1.2, 0.8 };
    int salaryDemand[50] = {90000, 240000, 440000, 120000, 160000, 290000, 330000, 170000, 460000, 370000, 490000, 230000, 300000, 190000, 210000, 480000, 180000, 320000, 430000, 50000, 280000, 390000, 140000, 500000, 360000, 410000, 260000, 210000, 170000, 350000, 440000, 390000, 100000, 330000, 130000, 200000, 50000, 500000, 250000, 470000, 240000, 380000, 460000, 490000, 120000, 420000, 150000, 430000, 290000, 280000 };
    // User inputs for X and Y
    int X, Y;
    printf("Enter X: ");
    scanf("%d", &X);
    printf("Enter Y: ");
    scanf("%d", &Y);

    // Call the function
    printStatistics(data, GPA, salaryDemand, X, Y);

    return 0;
}
```

Question#3: **[24 points, CLO1, 25 mins]**

Let's consider a special game that involves rolling a die and keeping track of scores and messages. In this game:

- Instead of the usual 1-6 values, each die roll can produce any signed integer value.
- Players can make up to 50 consecutive rolls of the die.
- The game keeps track of a score and a game message.
- The current score is calculated as the sum of the values of the last two die rolls, which means it can change throughout the game.
- The initial game message is "begin," and its length can be at most 200 characters.

Your task is to write a C program to simulate this game. Your program should allow the following operations:

a) GamePlay:

- Whenever the score becomes positive, call the function void Message (char msg[200]). This function should append the string "pos" to the game message.
- Whenever the score becomes negative, call the function int MaxScore (int rolls[]). This function should return the maximum score seen in all the previous die rolls.

You need to write the definitions of these functions.

b) Simulating Rolls and Game Messages:

After completing 50 die rolls, you should call the function void Utility (int rolls[], char msg[200]). This function is designed as follows:

- It processes the results of the die rolls.
- For each set of five die rolls, it extracts and prints the last five characters from the game message.
- This process repeats for the first set of five rolls and continues in the same pattern.

Solution:

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

void Message(char msg[200])

{

   strcat(msg, "pos");

}

int MaxScore(int rolls[])

{

   int max = -1000; // could be any small -ve value

   for(int i = 0; i < 50; i++)

   {

      int score;
```

```
        if(i >= 1)
            score =  rolls[i] + rolls[i-1];


        if(max < (rolls[i] + rolls[i-1]))
            max = (rolls[i] + rolls[i-1]);
    }
return max;
}
void Utility(int rolls[], char msg[200])
{
    int str_length = strlen(msg);
    for(int i = 0; i < 50; i+= 5)
    {
        for(int j = i, k = str_length-1; j < i+5; j++, k--)
        {
            printf("%d", rolls[j]);
            printf("%c", msg[k]);
        }
    }
}
int main()
{
    int rolls[50];
    int score = 0;
    char msg[200] = "begin";
    for(int i = 0; i < 50; i++)
    {
        scanf("%d", &rolls[i]);
        if(i >= 1)
            score =  rolls[i] + rolls[i-1];
        if(score >= 0)
            Message(msg);
```

```
        else
            int max = MaxScore(rolls);
    }
    Utility(rolls, msg);
}
```

Rubrics:

Part a	Parameter passing 2	Concatenation 8	
Part b	Function call 4	Logic for max score 6	
Part c	Parameter passing 2	Correct loop logic to fetch five scores at a time 5	Correction loop logic to fetch last five characters at a time 5

----------BEST OF LUCK----------