SHRI GOVINDRAM SEKSARIA INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE

Department of Applied Mathematics and Computational Science

MCA I Year Semester I Session: July-December 2025

LAB ASSIGNMENT

MA10401: STATISTICAL COMPUTING TECHNIQUES

SUBMITTED TO	**SUBMITTED BY**
Dr. Deepti Mokati	Full name of Student : Rohan Yadav
Dr. Kajal Mittal	Enrollment No. : 0801CA251120

INDEX

Q1. Write a program to calculate mean , median , mode in an individual series.

CODE :

```
 #include <stdio.h>
#include <conio.h>
void sort(float arr[], int n) {
    int i, j;
    float temp;
    for(i=0; i<n-1; i++) {
        for(j=i+1; j<n; j++) {
            if(arr[i]>arr[j]) {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
void main() {
    int n, i, j, ucount=0, cf=0, maxCount=0, found;
    float mean, median, mode, sum_fx=0;
    float arr[50], unique[50];
    int freq[50];
    clrscr();
    printf("Enter total number of elements: ");
    scanf("%d", &n);
```

```
printf("Enter elements: ");
for(i=0; i<n; i++)
    scanf("%f", &arr[i]);
sort(arr, n);
// Frequency table calculation
for(i=0; i<n; i++) {
    found = 0;
    for(j=0; j<ucount; j++) {
        if(arr[i]==unique[j]) {
            freq[j]++;
            found = 1;
            break;
        }
    }
    if(!found) {
        unique[ucount] = arr[i];
        freq[ucount] = 1;
        ucount++;
    }
}
// Print frequency table
printf("\n------------------------------");
printf("\n|  x  |  f  |  cf  |  fx  |");
printf("\n------------------------------");
```

```
cf = 0;

sum_fx = 0;

for(i=0; i<ucount; i++) {

    cf += freq[i];

    sum_fx += unique[i]*freq[i];

    printf("\n| %4.1f | %3d | %3d | %5.1f |", unique[i], freq[i], cf, unique[i]*freq[i]);

}

printf("\n------------------------------");


// Calculate mean

mean = sum_fx / n;


// Calculate median

if(n%2==0)

    median = (arr[n/2] + arr[(n/2)-1])/2;

else

    median = arr[n/2];


// Calculate mode

maxCount = 0;

for(i=0; i<ucount; i++) {

    if(freq[i] > maxCount) {

        maxCount = freq[i];

        mode = unique[i];

    }
    }
    printf("\nMean = %.2f", mean);
    printf("\nMedian = %.2f", median);
    printf("\nMode = %.2f", mode);
    printf("\nHighest Frequency = %d", maxCount);


    getch();
}
```

output:

```
------------------------------------------------
|  x   |  f  |  cf  |   fx   |
------------------------------------------------
|  2.00 |     2 |      2 |     4.00 |
|  3.00 |     1 |      3 |     3.00 |
|  4.00 |     3 |      6 |    12.00 |
|  5.00 |     1 |      7 |     5.00 |
|  6.00 |     1 |      8 |     6.00 |
------------------------------------------------
Mean = 4.00
Median = 4.00
Mode = 4.00
Highest Frequency = 3
```

2.Write a program to calculate mean , median , mode in a discrete series.

Code:

```
#include <stdio.h>
#include <conio.h>
void main() {
    int n, i, j, cf = 0, maxf = 0, mode_index = 0;
    float mean = 0, median = 0, mode = 0, sum_fx = 0;
   clrscr();
 printf("Enter total number of terms: ");
    scanf("%d", &n);
// Arrays for x and f
    float x[50];   // Values
    int f[50];     // Frequencies
    int cf_arr[50]; // Cumulative frequencies
    printf("\nEnter the values of x (Discrete values):\n");
    for (i = 0; i < n; i++) scanf("%f", &x[i]);
   printf("\nEnter the corresponding frequencies f:\n");
    for (i = 0; i < n; i++) scanf("%d", &f[i]);
   // Calculate cumulative frequency and fx
    printf("\n-----------------------------------------------------");
    printf("\n|   x   |   f   |   cf   |   f*x   |");
    printf("\n-----------------------------------------------------");
    for (i = 0; i < n; i++) {
        cf += f[i];
        cf_arr[i] = cf;
        sum_fx += x[i] * f[i];
        printf("\n| %5.2f | %5d | %5d | %7.2f |", x[i], f[i], cf_arr[i], x[i]*f[i]);
    }
  printf("\n--------------------------------------------------");
// Calculate Mean
    mean = sum_fx / cf;
 // Calculate Median
    // Find the class where cf >= N/2
    float Nby2 = cf / 2.0;
    for (i = 0; i < n; i++) {
        if (cf_arr[i] >= Nby2) {
            median = x[i];
            break;
        }
    }// Calculate Mode
    for (i = 0; i < n; i++) {
        if (f[i] > maxf) {
            maxf = f[i];
            mode_index = i;
        }
    }
    mode = x[mode_index];
    printf("\nTotal Σf = %d", cf);
    printf("\nTotal Σf*x = %.2f", sum_fx);
    printf("\n--------------------------------------------------");
    printf("\nMean = %.2f", mean);
    printf("\nMedian = %.2f", median);
    printf("\nMode = %.2f", mode);
    printf("\n-----------------------------------------------------");
getch();
}
```

```
-----------------------------------------------------
|  x   |  f  |  cf  |  f*x  |
-----------------------------------------------------
| 10.00 |     2 |     2 |   20.00 |
| 20.00 |     3 |     5 |   60.00 |
| 30.00 |     8 |    13 |  240.00 |
| 40.00 |     4 |    17 |  160.00 |
| 50.00 |     3 |    20 |  150.00 |
-----------------------------------------------------
Total Σf = 20
Total Σf*x = 630.00
-----------------------------------------------------
Mean = 31.50
Median = 30.00
Mode = 30.00
```

3.Write a program to calculate mean , median , mode in a continuous series.

Code:

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

int main()
{
    int i, n;
    float lower[50], upper[50], freq[50];
    float mid[50], cf[50];
    float sum_fx = 0, sum_f = 0;
    float mean, median, mode;
    float class_size;
    int median_class_index = 0, modal_class_index = 0;
    float N, cf_before, f_modal, f1, f2;

    clrscr();

    printf("Enter number of classes: ");
    scanf("%d", &n);

    printf("Enter lower and upper limits of each class:\n");
    for(i = 0; i < n; i++) {
        printf("Class %d lower limit: ", i + 1);
        scanf("%f", &lower[i]);
        printf("Class %d upper limit: ", i + 1);
        scanf("%f", &upper[i]);
    }

    printf("\nEnter frequencies for each class:\n");
    for(i = 0; i < n; i++) {
```

```
    printf("Frequency of class %d: ", i + 1);
    scanf("%f", &freq[i]);
}

// Calculate midpoints and cumulative frequencies
cf[0] = freq[0];
for(i = 0; i < n; i++) {
    mid[i] = (lower[i] + upper[i]) / 2.0;
    if(i > 0)
        cf[i] = cf[i-1] + freq[i];
    sum_fx += mid[i] * freq[i];
    sum_f += freq[i];
}

// Mean
mean = sum_fx / sum_f;

// Median
N = sum_f;
for(i = 0; i < n; i++) {
    if(cf[i] >= N/2) {
        median_class_index = i;
        break;
    }
}
class_size = upper[0] - lower[0];
cf_before = (median_class_index == 0) ? 0 : cf[median_class_index-1];
median = lower[median_class_index] + ((N/2 - cf_before) / freq[median_class_index]) * class_size;

// Mode
for(i = 0; i < n; i++) {
        if(freq[i] > freq[modal_class_index])
            modal_class_index = i;
    }
    f_modal = freq[modal_class_index];
    f1 = (modal_class_index == 0) ? 0 : freq[modal_class_index-1];
    f2 = (modal_class_index == n-1) ? 0 : freq[modal_class_index+1];
    mode = lower[modal_class_index] + ((f_modal - f1) / ((2*f_modal) - f1 - f2)) * class_size;

    // Print table
    printf("\n---------------------------------------------------------");
    printf("\n|  Class Interval  |  f  |   x   |  cf  |   fx   |");
    printf("\n---------------------------------------------------------");
    for(i = 0; i < n; i++) {
        printf("\n| %5.1f - %-5.1f  | %3.0f | %5.1f | %5.0f | %6.1f |",
               lower[i], upper[i], freq[i], mid[i], cf[i], mid[i]*freq[i]);
    }
    printf("\n---------------------------------------------------------");

    printf("\nMean   = %.2f", mean);
    printf("\nMedian = %.2f", median);
    printf("\nMode   = %.2f", mode);

    getch();
    return 0;
}
```

Output:

```
Enter number of classes: 5
Class 1 lower limit: 0
Class 1 upper limit: 10
Frequency of class 1: 5
Class 2 lower limit: 10
Class 2 upper limit: 20
Frequency of class 2: 8
Class 3 lower limit: 20
Class 3 upper limit: 30
Frequency of class 3: 12
Class 4 lower limit: 30
Class 4 upper limit: 40
Frequency of class 4: 7
Class 5 lower limit: 40
Class 5 upper limit: 50
Frequency of class 5: 3

-------------------------------------------------------
| Class Interval | f |   x   |  cf |   fx   |
-------------------------------------------------------
|  0.0 - 10.0    |  5 |  5.0 |     5 |  25.0 |
| 10.0 - 20.0    |  8 | 15.0 |    13 | 120.0 |
| 20.0 - 30.0    | 12 | 25.0 |    25 | 300.0 |
| 30.0 - 40.0    |  7 | 35.0 |    32 | 245.0 |
| 40.0 - 50.0    |  3 | 45.0 |    35 | 135.0 |
-------------------------------------------------------
Mean   = 23.71
Median = 22.50
Mode   = 22.50
```

Q4. Write a program to calculate geometric mean and harmonic mean for discrete series.

Code:

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

void main() {
    int n, i;
    float x[50], freq[50];
    float product = 1.0, sum_reciprocal = 0.0;
    float GM, HM, total_freq = 0;

    clrscr();

    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter values of x:\n");
    for(i = 0; i < n; i++)
        scanf("%f", &x[i]);

    printf("Enter corresponding frequencies f:\n");
    for(i = 0; i < n; i++)
        scanf("%f", &freq[i]);
    // Calculate product for GM and sum of reciprocals for HM
    for(i = 0; i < n; i++) {
        product *= pow(x[i], freq[i]);         // x^f for geometric mean
        sum_reciprocal += freq[i] / x[i];      // f/x for harmonic mean
        total_freq += freq[i];                 // sum of frequencies
    }

    // Geometric Mean
    GM = pow(product, 1.0 / total_freq);

    // Harmonic Mean
    HM = total_freq / sum_reciprocal;

    printf("\nGeometric Mean = %.4f", GM);
    printf("\nHarmonic Mean  = %.4f", HM);

    getch();
}
```

Output:

```
Enter number of elements: 3
Enter values of x: 2 4 8
Enter corresponding frequencies f: 3 2 1
Geometric Mean = 3.17
Harmonic Mean  = 3.00
```

5.Write a program to calculate mean deviation from mean , median , mode in continuous series.
Code:

```
#include <stdio.h>

#include <conio.h>

void main() {

   int n, i;

   float lower[50], upper[50], freq[50], mid[50], cf[50];

   float sum_fx=0, sum_f=0;

   float mean, median, mode;

   float class_size, cf_before, N;

   int median_class_index=0, modal_class_index=0;

   float f_modal, f1, f2;

   float MD_mean=0, MD_median=0, MD_mode=0;

   clrscr();

   printf("Enter number of classes: ");

   scanf("%d", &n);

   // Input class limits

   for(i=0; i<n; i++) {

      printf("Class %d lower limit: ", i+1);

      scanf("%f", &lower[i]);

      printf("Class %d upper limit: ", i+1);

      scanf("%f", &upper[i]);

   }    // Input frequencies

   printf("\nEnter frequencies for each class:\n");

   for(i=0; i<n; i++) {

      printf("Frequency of class %d: ", i+1);
```

```
        scanf("%f", &freq[i]);
    }
    // Calculate midpoints and cumulative frequency
    cf[0] = freq[0];
    for(i=0; i<n; i++) {
        mid[i] = (lower[i]+upper[i])/2.0;
        if(i>0) cf[i] = cf[i-1]+freq[i];
        sum_fx += mid[i]*freq[i];
        sum_f += freq[i];
    }
mean = sum_fx / sum_f;          // Mean
        // Median
    N = sum_f;
    for(i=0; i<n; i++) {
        if(cf[i]>=N/2) {
            median_class_index = i;
            break;
        }
    }
    class_size = upper[0]-lower[0];
    cf_before = (median_class_index==0)?0:cf[median_class_index-1];
    median = lower[median_class_index] + ((N/2 - cf_before)/freq[median_class_index])*class_size;
    // Mode
    for(i=0; i<n; i++)     if(freq[i] > freq[modal_class_index]) modal_class_index = i;
        f_modal = freq[modal_class_index];
```

```
 f1 = (modal_class_index==0)?0:freq[modal_class_index-1];
 f2 = (modal_class_index==n-1)?0:freq[modal_class_index+1];
 mode = lower[modal_class_index] + ((f_modal-f1)/((2*f_modal)-f1-f2))*class_size;

 // Mean Deviation calculations
 for(i=0; i<n; i++) {
    MD_mean += freq[i] * ((mid[i]-mean>0)?(mid[i]-mean):-(mid[i]-mean));
    MD_median += freq[i] * ((mid[i]-median>0)?(mid[i]-median):-(mid[i]-median));
    MD_mode += freq[i] * ((mid[i]-mode>0)?(mid[i]-mode):-(mid[i]-mode));
 }
 MD_mean /= sum_f;
 MD_median /= sum_f;
 MD_mode /= sum_f;

// Print table
 printf("\n---------------------------------------------------------");
 printf("\n| Class Interval |  f  |  x  |  cf  |  fx  |");
 printf("\n---------------------------------------------------------");
 for(i=0; i<n; i++) {
    printf("\n| %4.1f-%4.1f | %3.0f | %4.1f | %3.0f | %5.1f |",
           lower[i], upper[i], freq[i], mid[i], cf[i], mid[i]*freq[i]);
 }
 printf("\n---------------------------------------------------------");

 printf("\nMean = %.2f", mean);
```

```
    printf("\nMedian = %.2f", median);
    printf("\nMode = %.2f", mode);
    printf("\nMean Deviation from Mean = %.2f", MD_mean);
    printf("\nMean Deviation from Median = %.2f", MD_median);
    printf("\nMean Deviation from Mode = %.2f", MD_mode);
 getch();
}
```

Output:

```
Enter number of classes: 5
Class 1 lower limit: 0
Class 1 upper limit: 10
Frequency of class 1: 5
Class 2 lower limit: 10
Class 2 upper limit: 20
Frequency of class 2: 8
Class 3 lower limit: 20
Class 3 upper limit: 30
Frequency of class 3: 12
Class 4 lower limit: 30
Class 4 upper limit: 40
Frequency of class 4: 7
Class 5 lower limit: 40
Class 5 upper limit: 50
Frequency of class 5: 3
------------------------------------------------------
```

```
| Class Interval |  f  |  x  |  cf  |  fx  |
----------------------------------------------------------
| 0.0-10.0 |  5 |  5.0 |  5 |  25.0 |
| 10.0-20.0 |  8 | 15.0 | 13 | 120.0 |
| 20.0-30.0 | 12 | 25.0 | 25 | 300.0 |
| 30.0-40.0 |  7 | 35.0 | 32 | 245.0 |
| 40.0-50.0 |  3 | 45.0 | 35 | 135.0 |
----------------------------------------------------------
Mean = 23.71
Median = 22.50
Mode = 22.50
Mean Deviation from Mean = 9.11
Mean Deviation from Median = 9.06
Mean Deviation from Mode = 9.06
```

Q6.Write a program to calculate standard deviation for continuous series using array.

Code:

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

void main() {
   int n, i;
   float lower[50], upper[50], freq[50], mid[50];
   float sum_fx=0, sum_f=0, mean, SD, sum_sq=0;
   float class_size;

   clrscr();

   printf("Enter number of classes: ");
   scanf("%d", &n);

   // Input class limits
   for(i=0; i<n; i++) {
      printf("Class %d lower limit: ", i+1);
      scanf("%f", &lower[i]);
      printf("Class %d upper limit: ", i+1);
      scanf("%f", &upper[i]);
   }

   // Input frequencies
   printf("\nEnter frequencies for each class:\n");
   for(i=0; i<n; i++) {
      printf("Frequency of class %d: ", i+1);
      scanf("%f", &freq[i]);
   }
```

```
    // Calculate midpoints and total f*x
    for(i=0; i<n; i++) {
        mid[i] = (lower[i]+upper[i])/2.0;
        sum_fx += mid[i]*freq[i];
        sum_f += freq[i];
    }

    mean = sum_fx / sum_f;    // Mean

    // Calculate sum of squared deviations
    for(i=0; i<n; i++) {
        sum_sq += freq[i] * (mid[i] - mean) * (mid[i] - mean);
    }

    // Standard Deviation
    SD = sqrt(sum_sq / sum_f);  // population SD
    // For sample SD, use: SD = sqrt(sum_sq / (sum_f-1));

    // Print frequency table
    printf("\n-------------------------------------------");
    printf("\n| Class Interval |  f  |  x  |  f*x  |");
    printf("\n-------------------------------------------");
    for(i=0; i<n; i++) {
        printf("\n| %4.1f-%4.1f | %3.0f | %4.1f | %5.1f |",
               lower[i], upper[i], freq[i], mid[i], mid[i]*freq[i]);
    }
    printf("\n-------------------------------------------");
    printf("\nMean = %.2f", mean);
    printf("\nStandard Deviation = %.2f", SD);
    getch();
}
```

Input:

```
Enter number of classes: 5
Class 1 lower limit: 0
Class 1 upper limit: 10
Frequency of class 1: 5
Class 2 lower limit: 10
Class 2 upper limit: 20
Frequency of class 2: 8
Class 3 lower limit: 20
Class 3 upper limit: 30
Frequency of class 3: 12
Class 4 lower limit: 30
Class 4 upper limit: 40
Frequency of class 4: 7
Class 5 lower limit: 40
Class 5 upper limit: 50
Frequency of class 5: 3
```

Output:

```
------------------------------------------
| Class Interval |  f  |  x  |  f*x  |
------------------------------------------
| 0.0-10.0 |  5 |  5.0 |  25.0 |
| 10.0-20.0 |  8 | 15.0 | 120.0 |
| 20.0-30.0 | 12 | 25.0 | 300.0 |
| 30.0-40.0 |  7 | 35.0 | 245.0 |
| 40.0-50.0 |  3 | 45.0 | 135.0 |
------------------------------------------
Mean = 23.71
Standard Deviation = 11.68
```

Q7.Write a program to calculate one period ahead forecast using Naïve method.

Code:

```
#include <stdio.h>
#include <conio.h>
void main() {
    int n, i;
    float actual[50], forecast[50];
    clrscr();
    printf("Enter number of periods: ");
    scanf("%d", &n);
   printf("Enter actual values:\n");
    for(i = 0; i < n; i++) {
        printf("Period %d: ", i+1);
        scanf("%f", &actual[i]);
    }
    printf("\nPeriod\tActual\tForecast");       // Naive Forecast: Forecast for t+1 = Actual at t
    printf("\n----------------------------");
    forecast[0] = 0; // No forecast for first period
    for(i = 0; i < n; i++) {
        if(i == 0)
            printf("\n%d\t%.2f\t--", i+1, actual[i]);
        else {
            forecast[i] = actual[i-1];
            printf("\n%d\t%.2f\t%.2f", i+1, actual[i], forecast[i]);
        }
    }
       float next_forecast = actual[n-1];     // One period ahead forecast
    printf("\n\nOne period ahead forecast (Period %d) = %.2f", n+1, next_forecast);

    getch();
}
```

Input:

```
Enter number of periods: 5
Period 1: 120
Period 2: 135
Period 3: 150
Period 4: 145
Period 5: 160
```

Output:

```
Period  Actual  Forecast
----------------------------
1       120     --
2       135     120
3       150     135
4       145     150
5       160     145

One period ahead forecast (Period 6) = 160
```

Q11.Write a program to generate random numbers using mid square method.

CODE :

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

int midSquare(int seed) {
   long long square = (long long)seed * seed;  // Square the seed
   int numDigits = 0;
   long long temp = seed;

      while (temp > 0) {
      numDigits++;
      temp /= 10;
   }

   int removeDigits = numDigits / 2;
   long long divisor = pow(10, removeDigits);
   long long middle = (square / divisor) % (long long)pow(10, numDigits);

   return (int)middle;
}

int main() {
   int seed, n, i;

   printf("Enter the seed value: ");
   scanf("%d", &seed);

   printf("Enter how many random numbers to generate: ");
   scanf("%d", &n);

   printf("\nRandom numbers generated using Mid-Square Method:\n");
   for (i = 0; i < n; i++) {
      seed = midSquare(seed);
      printf("%d\n", seed);
   }

   return 0;
}
```

OUTPUT :

```
Enter number of random numbers to generate: 5
Enter the seed value: 1234

Random Numbers Generated :

Random number 1: 5227
Random number 2: 3216
Random number 3: 3442
Random number 4: 5585
Random number 5: 1962
```