Shri G.S. Institute of Technology and Science

INODRE

Department of Applied Mathematics & Computational Science

MCA First Year Semester I July-December 2024

Lab Assignment

MA 10210: STATISTICAL COMPUTING TECHNIQUES

SUBMITED TO

Dr. Deepti Mokati

Ms. Payal Khandelwal

SUBMITTED BY

SHIV ARORA

Enrolment No.

0801CA241133

INDEX

Program 1. Write a program to calculate mean, median and mode in an individual series.

```
#include<stdio.h>
float mean(int num, int *arr){
   float sum = 0;
   for(int i=0; i<num; i++){
      sum += arr[i];}
   float meann = sum / num;
   return meann;}
float median(int num, int *arr){
   float ans;
   for(int i=0; i<num; i++){
      for(int j=i+1; j<num; j++){
         if(arr[i]> arr[j]){
            int a = arr[i];
            arr[i] = arr[j];
            arr[j] = a;} } }
   if(num % 2 == 0){
      int one = num / 2;
      int two = one - 1;
      float x = arr[one];
      float y = arr[two];
      ans = (x + y) / 2;
   }else ans = arr[((num + 1) / 2) - 1];
   return ans;}
int mode(int num, int *arr){
   int max_Value = 0;
   int max_Count = 0;
   for (int i = 0; i < num; i++){
      int cnt = 0;
      for (int j = 0; j < num; j++){
```

```
            if (arr[j] == arr[i])
                cnt++;}
            if (cnt > max_Count){
                max_Count = cnt;
                max_Value = arr[i];}}
        return max_Value;}
    void main(){
        int num;
        printf("Enter the number of observations  ");
        scanf("%d", &num);
        int arr[num];
        printf("Enter the values of the observation\n");
        for(int i=0; i<num; i++){
            scanf("%d", &arr[i]);}
        float Mean = mean(num, arr);
        float Median = median(num, arr);
        int Mode = mode(num, arr);
        printf("\nMean = %.2f\tMedian = %.2f\tMode = %d", Mean, Median, Mode);
    }
```

OUTPUT:

```
\ucrt64\bin\gdb.exe   --interpreter=mi
Enter the number of observations  10
Enter the values of the observation
25 85 39 47 55 68 73 43 20 25

Mean = 48.00    Median = 45.00  Mode = 25
PS D:\Stats>
```

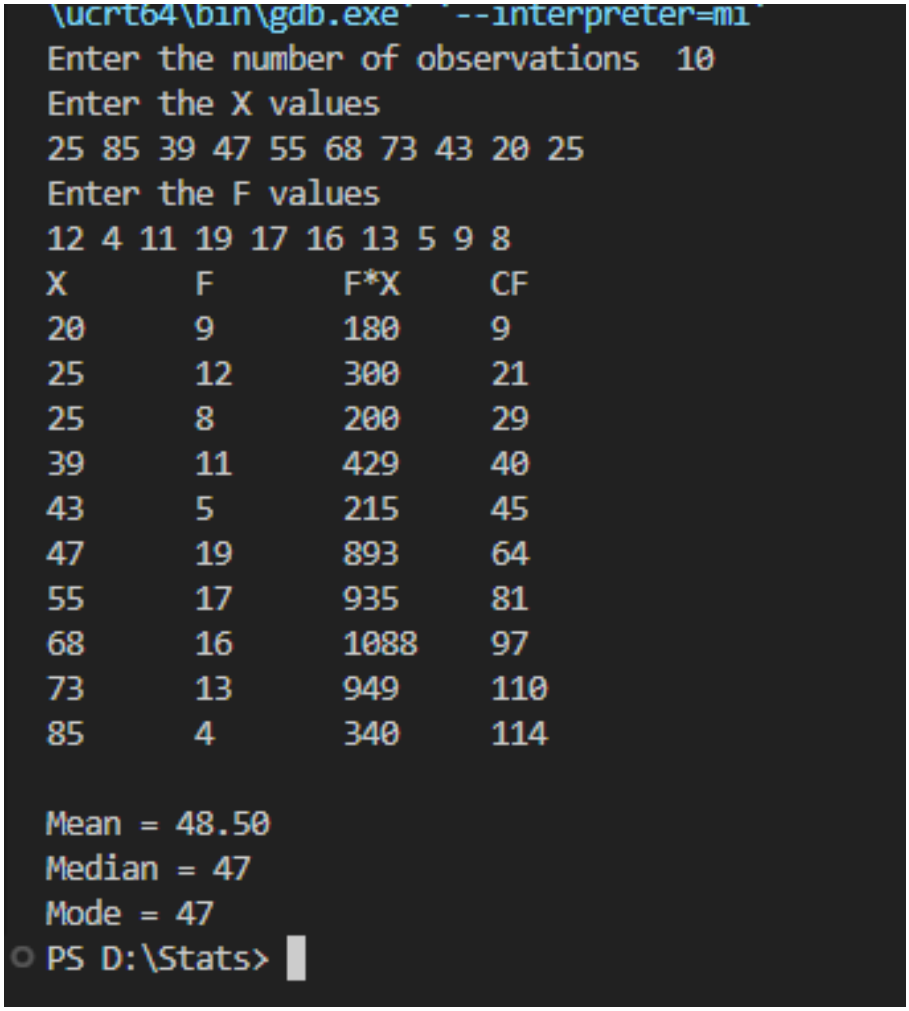
Program 2. Write a program to calculate mean, median and mode in a discrete series.

```
#include<stdio.h>
void main(){
   int size;
   printf("Enter the number of observations  ");
   scanf("%d", &size);
   int arrX[size];
   printf("Enter the X values\n");
   for(int i=0; i<size; i++){
      scanf("%d", &arrX[i]);
   }
   int arrF[size];
   printf("Enter the F values\n");
   for(int i=0; i<size; i++){
      scanf("%d", &arrF[i]);
   }
   // Sorting the X and Y table
   for(int i=0; i<size; i++){
      for(int j=i+1; j<size; j++){
         if(arrX[i]> arrX[j]){
            int x = arrX[i];
            int f  = arrF[i];

            arrX[i] = arrX[j];
            arrF[i] = arrF[j];

            arrX[j] = x;
            arrF[j] = f;
   } } }
   // Making of TABLE
```

```
int FX[size], CF[size];
int add = 0;
for(int i=0; i<size; i++){
   FX[i] = arrX[i] * arrF[i];
   add += arrF[i];
   CF[i] = add;

}
// Printing the table
printf("X\tF\tF*X\tCF\n");
for(int i=0; i<size; i++){
   printf("%d\t%d\t%d\t%d\n", arrX[i], arrF[i], FX[i], CF[i]);
}
//  MEAN
float sumFX = 0;
float sumF = 0;
for(int i=0; i<size; i++){
   sumFX += FX[i];
   sumF += arrF[i];
}
float mean = sumFX / sumF;
printf("\nMean = %.2f\n", mean);
// MEDIAN
float term = (sumF + 1) / 2;
int position = 0;
for(int i=0; i<size; i++){
   if(CF[i] < term){
      continue;
   }else{
      position = i;
```

```
            break;
        }}
    printf("Median = %.d\n", arrX[position]);
    // MODE
    int highest = 0;
    int pos = 0;
    for(int i=0; i< size; i++){
        if(arrF[i] > highest) {
            highest = arrF[i];
            pos = i;
        }
    }
    printf("Mode = %d\n", arrX[pos]);
}
```

OUTPUT:


\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter the number of observations 10
Enter the X values
25 85 39 47 55 68 73 43 20 25
Enter the F values
12 4 11 19 17 16 13 5 9 8
X F F*X CF
20 9 180 9
25 12 300 21
25 8 200 29
39 11 429 40
43 5 215 45
47 19 893 64
55 17 935 81
68 16 1088 97
73 13 949 110
85 4 340 114
Mean = 48.50
Median = 47
Mode = 47
PS D:\Stats>

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

```
#include<stdio.h>
void main(){
   int size;
   printf("Enter the number of observations  ");
   scanf("%d", &size);
   int upperB[size];
   int lowerB[size];
   printf("Enter the upper and lower bound values\n");
   for(int i=0; i<size; i++){
      scanf("%d%d", &lowerB[i], &upperB[i]);
   }
   int arrF[size];
   printf("Enter the F values\n");
   for(int i=0; i<size; i++){
      scanf("%d", &arrF[i]);
   }
   // Making the table
   int mid_value[size];
   int FM[size];
   int CF[size];
   int add = 0;
   for(int i=0; i< size; i++){
      mid_value[i] = (upperB[i] + lowerB[i]) / 2;
      FM[i] = mid_value[i] * arrF[i];
      add += arrF[i];
      CF[i] = add;
   }
   // Printing the table
   printf("intervals\tFreq\tmid values(M)\tF*M\tCF\n");
```

```
    for(int i=0; i<size; i++){
        printf("%d-%d\t\t%d\t\t%d\t%d\t%d\n",lowerB[i], upperB[i], arrF[i],
mid_value[i], FM[i], CF[i]);
    }
    // //  MEAN
    float sumFX = 0;
    float sumF = 0;
    for(int i=0; i<size; i++){
        sumFX += FM[i];
        sumF += arrF[i];

    }
    float mean = sumFX / sumF;
    printf("\nMean = %.2f\n", mean);
    // MEDIAN
    float term = sumF / 2;
    int positionCF = 0;
    int positionF = 0;
    for(int i=0; i<size; i++){
        if(CF[i] < term){
            int a;
        }else{
            positionF = i;
            positionCF = i - 1;
            break;
        }
    }
    float x = (term - CF[positionCF])/ arrF[positionF];
    float y = (upperB[positionF] - lowerB[positionF]);
    float z = lowerB[positionF];
    printf("Median = %.2f\n", x * y + z);
```

```
    // // MODE
    int highest = 0;
    int f1, f2 , f0 = 0;
    for(int i=0; i< size; i++){
       if(arrF[i] > highest) {
          highest = arrF[i];
          f1 = i;
       }
    }
    if(arrF[f1] == (size - 1)){
       f0 = f1 - 1;
       f2 = 0;
    }else if(arrF[f1] == 0){
       f0 = 0;
       f2 = f1 + 1;
    }else{
       f0 = f1 -1;
       f2 = f1 + 1;
    }
    float a = upperB[positionF] - lowerB[positionF];
    float b = ((2 * arrF[f1]) - arrF[f0] - arrF[f2]);
    float c = (arrF[f1] - arrF[f0]);
    float d = lowerB[f1];
    float a1 =  c/ b;
    float a2 = a1 * a;
    printf("Mode =  %.2f", a2 + d);
}
```

OUTPUT:

```
Enter the number of observations  10
Enter the upper and lower bound values
0 10 10 20 20 30 30 40 40 50 50 60 60 70 70 80 80 90 90 100
Enter the F values
11 5 8 14 12 6 8 7 19 16
intervals       Freq    mid values(M)   F*M     CF
0-10            11              5       55      11
10-20           5               15      75      16
20-30           8               25      200     24
30-40           14              35      490     38
40-50           12              45      540     50
50-60           6               55      330     56
60-70           8               65      520     64
70-80           7               75      525     71
80-90           19              85      1615    90
90-100          16              95      1520    106

Mean = 55.38
Median = 55.00
Mode =  88.00
PS D:\Stats>
```

Program 4. Write a program to calculate geometric and harmonic mean in a discrete series.

```
#include<stdio.h>
#include<math.h>
void main(){
   int size;
   printf("ENTER the number of elements\n");
   scanf("%d", &size);
   // The X column
   int arrX[size];
   printf("ENTER X elements\n");
   for(int i=0; i<size; i++){
      scanf("%d", &arrX[i]);
   }
   // THe F Cloumn
   int arrF[size];
   printf("ENTER the Frequency\n");
   for(int i=0; i<size; i++){
      scanf("%d", &arrF[i]);
   }
   // Taking log values
   double logg[size];
   for(int i=0; i<size; i++){
      logg[i] = log(arrX[i]);
   }
   // Taking submission of F column
   int freq_sum = 0;
   for(int i=0; i<size; i++){
      freq_sum += arrF[i];
   }
   // Making f/x column
```

```
    double arrFX[size];
    for(int i=0; i<size; i++){
        arrFX[i] = (double)arrF[i] / (double)arrX[i];
    }
    // Printing the table
    printf("\n X \t F \t log(x) \t f/x\n");
    for(int i=0; i<size; i++){
        printf("%d \t %d \t %.4lf \t %.4lf\n", arrX[i], arrF[i], logg[i], arrFX[i]);
    }
    //  ---------------------------------- Calculating G.M ----------------------------------
    // Taking Log array sum
    double log_sum = 0;
    for(int i=0; i<size; i++){
        log_sum += logg[i];
    }
    // Taking antilog
    double result_gm  = exp(log_sum / size);
    printf("Geometric Mean = %.3lf\n", result_gm);
    // ---------------------------------- Calculating H.M ----------------------------------
    // Taking f/x array sum
    double fx_sum = 0;
    for(int i=0; i<size; i++){
        fx_sum += arrFX[i];
    }
    // Calculating sum
    double result_hm = freq_sum / fx_sum;
    printf("Harmonix mean Mean = %.3lf", result_hm);
}
```

OUTPUT:

```
ENTER the number of elements
10
ENTER X elements
70 65 15 8 10 22 28 30 34 7
ENTER the Frequency
2 5 8 10 7 4 4 12 9 3

 X        F        log(x)          f/x
70        2        4.2485          0.0286
65        5        4.1744          0.0769
15        8        2.7081          0.5333
8         10       2.0794          1.2500
10        7        2.3026          0.7000
22        4        3.0910          0.1818
28        4        3.3322          0.1429
30        12       3.4012          0.4000
34        9        3.5264          0.2647
7         3        1.9459          0.4286
Geometric Mean = 21.779
Harmonix mean Mean = 15.973
PS D:\Stats>
```

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

```
#include<stdio.h>
#include <math.h>
void main(){
   int size;
   printf("Enter the number of elements\n");
   scanf("%d", &size);
   int upper[size], lower[size];
   printf("Enter the upper and lower bound values\n");
   for(int i=0; i<size; i++){
      scanf("%d%d", &lower[i], &upper[i]);
   }
   int arrF[size];
   printf("Enter the frequency values\n");
   for(int i=0; i<size; i++){
      scanf("%d", &arrF[i]);
   }
   // Mid value
   int arrX[size];
   for(int i=0; i<size; i++){
      arrX[i] = (upper[i] + lower[i]) /  2;
   }
   // cf
   int arrCF[size];
   int temp_sum= 0;
   for(int i=0; i<size; i++){
      temp_sum += arrF[i];
      arrCF[i] = temp_sum;
   }
   // fx
```

```
int arrFX[size];
for(int i=0; i<size; i++){
   arrFX[i] = arrX[i] * arrF[i];
}
// Mean
int  sumFX=0, sumF=0;
for(int i=0; i<size; i++){
   sumF += arrF[i];
   sumFX += arrFX[i];
}
float mean = (float)sumFX / (float) sumF;
// Median
float term = sumF / 2;
int positionCF = 0;
int positionF = 0;
for(int i=0; i<size; i++){
   if(arrCF[i] < term){
      int a;
   }else{
      positionF = i;
      positionCF = i - 1;
      break;
   }
}
float x = (term - arrCF[positionCF])/ arrF[positionF];
float y = (upper[positionF] - lower[positionF]);
float z = lower[positionF];
float median =  x * y + z;

// // MODE
```

```
int highest = 0;
int f1, f2 , f0 = 0;
for(int i=0; i< size; i++){
   if(arrF[i] > highest) {
      highest = arrF[i];
      f1 = i;
   }
}
if(arrF[f1] == (size - 1)){
   f0 = f1 - 1;
   f2 = 0;
}else if(arrF[f1] == 0){
   f0 = 0;
   f2 = f1 + 1;
}else{
   f0 = f1 -1;
   f2 = f1 + 1;
}
float a = upper[positionF] - lower[positionF];
float b = ((2 * arrF[f1]) - arrF[f0] - arrF[f2]);
float c = (arrF[f1] - arrF[f0]);
float d = lower[f1];
float a1 =  c/ b;
float a2 = a1 * a;
float mode = a2 + d;
//  |x - mean|
float xbar[size];
for(int i=0; i<size; i++){
   xbar[i] = fabs((float)arrX[i] - mean);
}
```

```
    // f * |x - mean|
    float xbarf[size];
    for(int i=0; i<size; i++){
        xbarf[i] = (float)xbar[i] * (float)arrF[i];
    }
    // |x - median|
    float median_xbar[size];
    for(int i=0; i<size; i++){
        median_xbar[i] = fabs((float)arrX[i] - median);
    }
    // f * |x - median|
    float median_xbarf[size];
    for(int i=0; i<size; i++){
        median_xbarf[i] = (float)median_xbar[i] * (float)arrF[i];
    }
    // |x - mode|
    float mode_xbar[size];
    for(int i=0; i<size; i++){
        mode_xbar[i] = fabs((float)arrX[i] - mode);
    }
    // f * |x - mode|
    float mode_xbarf[size];
    for(int i=0; i<size; i++){
        mode_xbarf[i] = (float)mode_xbar[i] * (float)arrF[i];
    }
    // Print Table
    printf("Interval\tfreq\tX\tCF\tf*x\t|x-mean|\tf*|x-mean|\t|x-median|\tf*|x-
median|\t|x-mode|\tf*|x-mode|\n");
    for(int i=0; i<size; i++){
        printf("%d-
%d\t\t%d\t%d\t%d\t%d\t%.2f\t\t%.2f\t\t%.2f\t\t%.2f\t\t%.2f\t\t%.2f\n",lower[i],
```

```
upper[i], arrF[i], arrX[i], arrCF[i], arrFX[i], xbar[i], xbarf[i], median_xbar[i],
median_xbarf[i], mode_xbar[i], mode_xbarf[i]);
    }
    // Meadian Deviation from Mean
    float MEAN;
    for(int i=0;i< size; i++){
       MEAN += xbarf[i]; }
    printf("\nMean deviation from MEAN is %.3f", MEAN/ (float)sumF);
    // Meadian Deviation from Median
    float MEDAIN;
    for(int i=0;i< size; i++){
       MEDAIN += median_xbarf[i];}
    printf("\nMean deviation from MEDAIN is %.3f", MEDAIN/ (float)sumF);
    // Meadian Deviation from Mean
    float MODE;
    for(int i=0;i< size; i++){
       MODE += mode_xbarf[i];  }
    printf("\nMean deviation from MODE is %.3f", MODE/ (float)sumF);}
```

OUTPUT:

```
Enter the number of elements
10
Enter the upper and lower bound values
0 10 10 20 20 30 30 40 40 50 50 60 60 70 70 80 80 90 90 100
Enter the frequency values
2 5 8 10 7 4 4 12 9 3
```

Interval	freq	X	CF	f*x	|x-mean|	f*|x-mean|	|x-median|	f*|x-median|	|x-mode|	f*|x-mode|
0-10	2	5	2	10	47.81	95.63	45.00	90.00	72.27	144.55
10-20	5	15	7	75	37.81	189.06	35.00	175.00	62.27	311.36
20-30	8	25	15	200	27.81	222.50	25.00	200.00	52.27	418.18
30-40	10	35	25	350	17.81	178.13	15.00	150.00	42.27	422.73
40-50	7	45	32	315	7.81	54.69	5.00	35.00	32.27	225.91
50-60	4	55	36	220	2.19	8.75	5.00	20.00	22.27	89.09
60-70	4	65	40	260	12.19	48.75	15.00	60.00	12.27	49.09
70-80	12	75	52	900	22.19	266.25	25.00	300.00	2.27	27.27
80-90	9	85	61	765	32.19	289.69	35.00	315.00	7.73	69.55
90-100	3	95	64	285	42.19	126.56	45.00	135.00	17.73	53.18

```
Mean deviation from MEAN is 23.125
Mean deviation from MEDAIN is 23.125
Mean deviation from MODE is 28.295
PS D:\Stats>
```

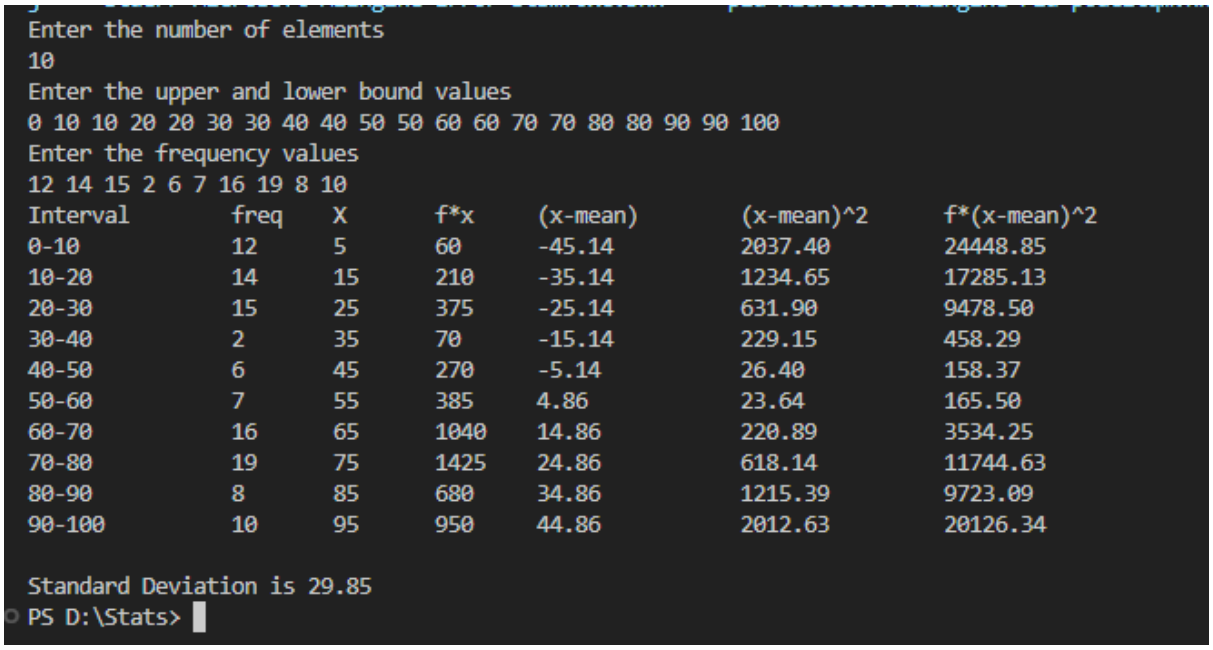
Program 6. Write a program to calculate standard deviation for continuous series using array.

```
#include<stdio.h>
#include <math.h>
void main(){
   int size;
   printf("Enter the number of elements\n");
   scanf("%d", &size);
   int upper[size], lower[size];
   printf("Enter the upper and lower bound values\n");
   for(int i=0; i<size; i++){
      scanf("%d%d", &lower[i], &upper[i]);
   }
   int arrF[size];
   printf("Enter the frequency values\n");
   for(int i=0; i<size; i++){
      scanf("%d", &arrF[i]);
   }
   // Mid value
   int arrX[size];
   for(int i=0; i<size; i++){
      arrX[i] = (upper[i] + lower[i]) /  2;
   }
   // fx
   int arrFX[size];
   for(int i=0; i<size; i++){
      arrFX[i] = arrX[i] * arrF[i];
   }
   // Mean
   int  sumFX=0, sumF=0;
   for(int i=0; i<size; i++){
```

```
    sumF += arrF[i];
    sumFX += arrFX[i];
  }
  float mean = (float)sumFX / (float) sumF;
  //  x - mean
  float xbar[size];
  for(int i=0; i<size; i++){
    xbar[i] = (float)arrX[i] - mean;
  }
  //  x - mean ^ 2
  float square_xbar[size];
  for(int i=0; i<size; i++){
    square_xbar[i] = xbar[i] * xbar[i];
  }
  // f * (x - mean)^2
  float xbarf[size];
  for(int i=0; i<size; i++){
    xbarf[i] = (float)square_xbar[i] * (float)arrF[i];
  }
  // Print Table
  printf("Interval\tfreq\tX\tf*x\t(x-mean)\t(x-mean)^2\tf*(x-mean)^2\n");
  for(int i=0; i<size; i++){
    printf("%d-%d\t\t%d\t%d\t%d\t%.2f\t\t%.2f\t\t%.2f\n",lower[i], upper[i], arrF[i],
arrX[i], arrFX[i], xbar[i], square_xbar[i], xbarf[i]);
  }
  // Standard Deviation
  float sum_xbarf;
  for(int i=0; i< size; i++){
    sum_xbarf += xbarf[i];
  }
  float SD = sqrt(sum_xbarf / sumF);
```

```
        printf("\nStandard Deviation is %.2f", SD);
    }
```

OUTPUT:



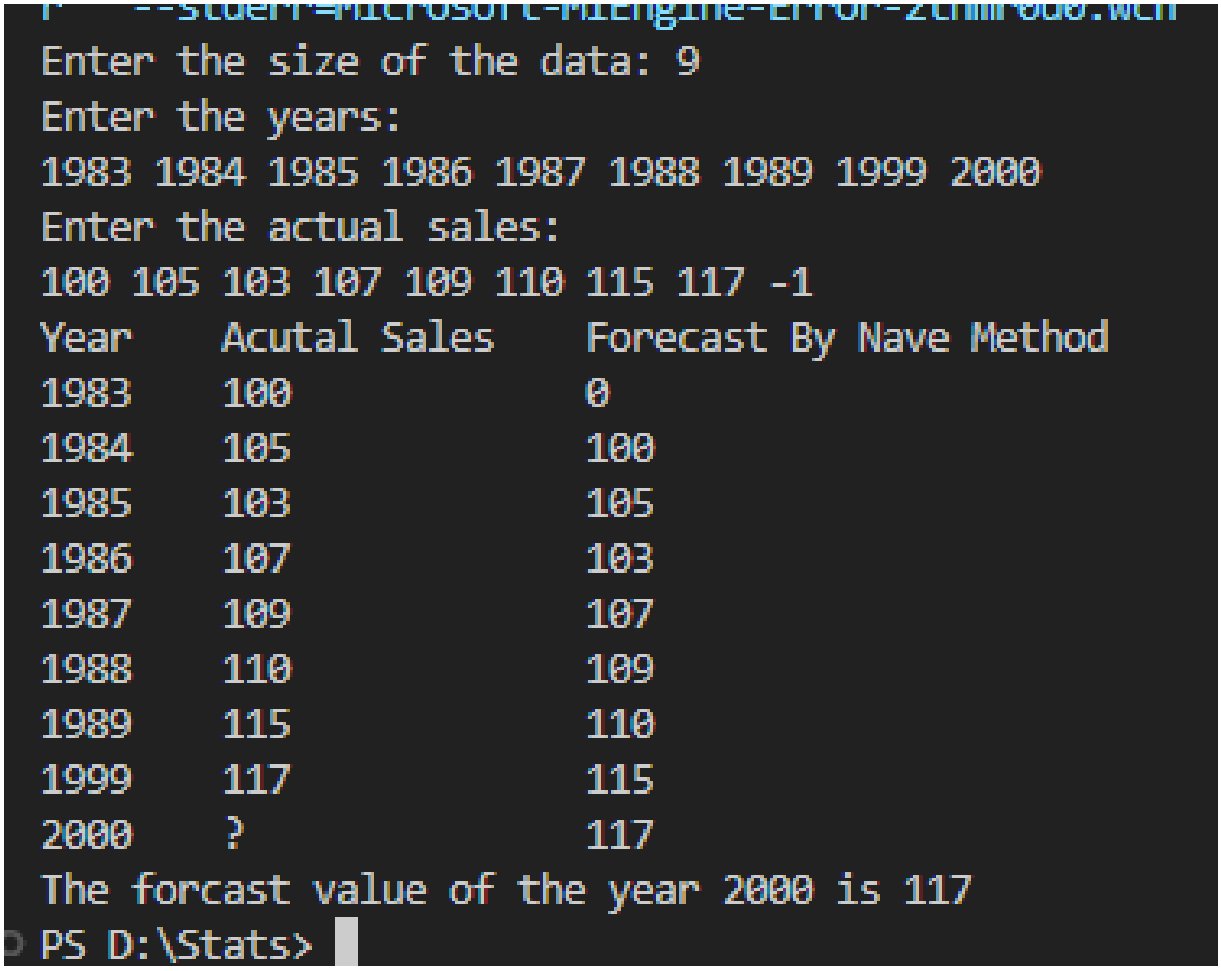
```
Enter the number of elements
10
Enter the upper and lower bound values
0 10 10 20 20 30 30 40 40 50 50 60 60 70 70 80 80 90 90 100
Enter the frequency values
12 14 15 2 6 7 16 19 8 10
Interval        freq    X       f*x     (x-mean)        (x-mean)^2      f*(x-mean)^2
0-10            12      5       60      -45.14          2037.40         24448.85
10-20           14      15      210     -35.14          1234.65         17285.13
20-30           15      25      375     -25.14          631.90          9478.50
30-40           2       35      70      -15.14          229.15          458.29
40-50           6       45      270     -5.14           26.40           158.37
50-60           7       55      385     4.86            23.64           165.50
60-70           16      65      1040    14.86           220.89          3534.25
70-80           19      75      1425    24.86           618.14          11744.63
80-90           8       85      680     34.86           1215.39         9723.09
90-100          10      95      950     44.86           2012.63         20126.34

Standard Deviation is 29.85
PS D:\Stats>
```

Program 7. Write a program to calculate one period ahead forecast using Naïve method.

```
#include<stdio.h>
void main(){
   int size;
   printf("Enter the size of the data: ");
   scanf("%d", &size);
   int year[size];
   int  actual_sales[size];

   printf("Enter the years:\n");
   for(int i=0; i<size; i++){
      scanf("%d", &year[i]);
   }
   printf("Enter the actual sales:\n");
   for(int i=0; i<size; i++){
      scanf("%d", &actual_sales[i]);
   }
   actual_sales[size-1] = -1;

   int forecast_by_nave_method[size];
   forecast_by_nave_method[0] = 0;
   for(int i=1; i<size; i++){
      forecast_by_nave_method[i] = actual_sales[i-1];
   }
   // Printing the table
   printf("Year\tAcutal Sales\tForecast By Nave Method\n");
   for(int i=0; i<size; i++){
      if (actual_sales[i] == -1){
         printf("%d\t?\t\t%d\n", year[i], forecast_by_nave_method[i]);
      }else{
```

```
            printf("%d\t%d\t\t%d\n", year[i], actual_sales[i],
forecast_by_nave_method[i]);
        }
    }
    printf("The forcast value of the year %d is %d", year[size-1],
forecast_by_nave_method[size-1]);
}
```

OUTPUT:



Program 8. Write a program to calculate one period ahead forecast using moving average method.

```
#include<stdio.h>
void main(){
   int size;
   printf("Enter the size of the data: ");
   scanf("%d", &size);
   int year[size];
   int  actual_sales[size];

   printf("Enter the years:\n");
   for(int i=0; i<size; i++){
      scanf("%d", &year[i]);
   }
   printf("Enter the actual sales:\n");
   for(int i=0; i<size; i++){
      scanf("%d", &actual_sales[i]);
   }
   actual_sales[size-1] = -1;

   int forecast_by_nave_method[size];
   forecast_by_nave_method[0] = 0;
   forecast_by_nave_method[1] = 0;
   forecast_by_nave_method[2] = 0;
   for(int i=3; i<size; i++){
      forecast_by_nave_method[i] = actual_sales[i-1] + actual_sales[i-2] +
actual_sales[i-3];
   }
   // Printing the table
   printf("Year\tAcutal Sales\tForecast By Nave Method\n");
```

```
    for(int i=0; i<size; i++){
        if (actual_sales[i] == -1){
            printf("%d\t?\t\t%d\n", year[i], forecast_by_nave_method[i]);
        }else{
            printf("%d\t%d\t\t%d\n", year[i], actual_sales[i],
forecast_by_nave_method[i]);
        }
    }
    printf("The forcast value of the year %d is %d", year[size-1],
forecast_by_nave_method[size-1]);
}
```

OUTPUT:

```
Enter the size of the data: 9
Enter the years:
1983 1984 1985 1986 1987 1988 1989 1999 2000
Enter the actual sales:
100 105 103 107 109 110 115 117 -1
Year    Acutal Sales    Forecast By Nave Method
1983    100             0
1984    105             0
1985    103             0
1986    107             308
1987    109             315
1988    110             319
1989    115             326
1999    117             334
2000    ?               342
The forcast value of the year 2000 is 342
PS D:\Stats>
```

Program 9. Write a program to calculate standard statistical measures using exponential smoothing forecasting method. [Mean Error (ME), Mean Absolute Error (MAE), Mean Square Error (MSE)]

```
#include<stdio.h>
#include<math.h>
void main(){
   int size;
   printf("Enter the size of the duration: ");
   scanf("%d", &size);
   int demand[size];
   printf("Enter the demand:\n");
   for(int i=0; i<size; i++){
      scanf("%d", &demand[i]);
   }
   float alpha;
   printf("Enter the alpha value:\n");
   scanf("%f", &alpha);
   float intial;
   printf("Enter the intial value of the forcaste:\n");
   scanf("%f", &intial);
   // Forcaste table
   float forcaste[size];
   forcaste[0]= intial;
   for(int i=1; i<size; i++){
      forcaste[i] =  forcaste[i-1] + ((float)alpha * (float)(demand[i-1] - forcaste[i-1]));
   }
   // Mean Error Table
   float me[size], mean_error = 0;
   for(int i=0; i<size; i++){
      me[i] = demand[i] - forcaste[i];
      mean_error += demand[i] - forcaste[i];
```

```
    }
    // Mean Square Error Table
    float mse[size], mean_square_error = 0;
    for(int i=0; i<size; i++){
        mse[i] = me[i] * me[i];
        mean_square_error += (me[i] * me[i]);
    }
    // Mean Absolute Error Table
    float mae[size], mean_absolute_error;
    for(int i=0; i<size; i++){
        mae[i] = fabs(demand[i] - forcaste[i]);
        mean_absolute_error += fabs(demand[i] - forcaste[i]);
    }
    // Print table
    printf("Duration\tDemand\tForcaste\tMeanError\tMeanSquareError\t  MeanAbsolut
eError\n");
    for (int i = 0; i < size; i++){
        printf("%d\t\t%d\t%f\t%f\t%f\t  %f\n", i+1, demand[i], forcaste[i], me[i], mse[i],
mae[i]);
    }
    float pred = forcaste[size-1] + alpha * (demand[size-1] - forcaste[size-1]);
    printf("%d\t\t0\t%f", size+1, pred);
    printf("\n");
    // Mean Errors
    printf("Mean Error = %.2f\n", mean_error / size);
    printf("Mean Absolute Error = %.2f\n", mean_absolute_error / size);
    printf("Mean Square Error = %.2f\n", mean_square_error / size);
    printf("Forcaste of 11th month = %.2f", pred);
}
```

OUTPUT:

```
Enter the size of the duration: 10
Enter the demand:
120 201 520 452 351 251 444 521 365 98
Enter the alpha value:
0.2
Enter the intial value of the forcaste:
200
Duration        Demand  Forcaste        MeanError       MeanSquareError   MeanAbsoluteError
1               120     200.000000      -80.000000      6400.000000       80.000000
2               201     184.000000      17.000000       289.000000        17.000000
3               520     187.399994      332.600006      110622.765625     332.600006
4               452     253.919998      198.080002      39235.687500      198.080002
5               351     293.536011      57.463989       3302.110107       57.463989
6               251     305.028809      -54.028809      2919.112061       54.028809
7               444     294.223053      149.776947      22433.134766      149.776947
8               521     324.178436      196.821564      38738.726563      196.821564
9               365     363.542755      1.457245        2.123563          1.457245
10              98      363.834198      -265.834198     70667.820313      265.834198
11              0       310.667358
Mean Error = 55.33
Mean Absolute Error = 135.31
Mean Square Error = 29461.05
Forcaste of 11th month = 310.67
PS D:\>
```

Program 10. Write a program to calculate relative measures of forecasting using exponential smoothing forecasting method. [Percentage Error (PE), Mean Percentage Error (MPE), mean Absolute Percentage Error (MAPE)]

```
#include<stdio.h>
#include<math.h>
void main(){
   int size;
   printf("Enter the size of the duration: ");
   scanf("%d", &size);
   int demand[size];
   printf("Enter the demand:\n");
   for(int i=0; i<size; i++){
      scanf("%d", &demand[i]);
   }
   float alpha;
   printf("Enter the alpha value:\n");
   scanf("%f", &alpha);
   float intial;
   printf("Enter the intial value of the forcaste:\n");
   scanf("%f", &intial);

   // Forcaste table
   float forcaste[size];
   forcaste[0]= intial;
   for(int i=1; i<size; i++){
      forcaste[i] =  forcaste[i-1] + ((float)alpha * (float)(demand[i-1] - forcaste[i-1]));
   }

   // Percent Error Table
   float pe[size];
   for(int i=0; i<size; i++){
```

```
        pe[i] = (float)((demand[i] - forcaste[i]) / (float)demand[i]) * 100.00;
    }

    // Mean percent Table
    float mpe[size], mean_percent_error = 0;
    for(int i=0; i<size; i++){
        mpe[i] = pe[i];
        mean_percent_error += pe[i] ;
    }

    // Mean Absolute Percent Error Table
    float mape[size], mean_percent_absolute_error;
    for(int i=0; i<size; i++){
        mape[i] = fabs(pe[i]);
        mean_percent_absolute_error += fabs(pe[i]);
    }

    // Print table
    printf("Duration\tDemand\tForcaste\tPercentError\t  MeanPercentAbsoluteError\n"
);
    for (int i = 0; i < size; i++){
        printf("%d\t\t%d\t%f\t%f\t \t %f\n", i+1, demand[i], forcaste[i], pe[i], mape[i]);
    }
    float pred = forcaste[size-1] + alpha * (demand[size-1] - forcaste[size-1]);
    printf("%d\t\t0\t%f", size+1, pred);
    printf("\n");

    // Mean Errors
    printf("Mean Percent Error = %.2f%%\n", mean_percent_error / size);
    printf("Mean Percent Absolute Error = %.2f%%\n", mean_percent_absolute_error /
size);
```

```
        printf("Forcaste of 11th month = %.2f", pred);
    }
```

OUTPUT:

```
Enter the size of the duration: 10
Enter the demand:
213 201 198 207 220 232 210 217 212 225
Enter the alpha value:
0.2
Enter the intial value of the forcaste:
208
Duration        Demand  Forcaste        PercentError      MeanPercentAbsoluteError
1               213     208.000000      2.347418                2.347418
2               201     209.000000      -3.980099               3.980099
3               198     207.399994      -4.747472               4.747472
4               207     205.519989      0.714981                0.714981
5               220     205.815994      6.447275                6.447275
6               232     208.652802      10.063448               10.063448
7               210     213.322235      -1.582017               1.582017
8               217     212.657791      2.001018                2.001018
9               212     213.526230      -0.719920               0.719920
10              225     213.220978      5.235121                5.235121
11              0       215.576782
Mean Percent Error = 1.58%
Mean Percent Absolute Error = 3.78%
Forcaste of 11th month = 215.58
PS D:\Stats>
```

Program 11. Write a program to generate random numbers using mid square method.

```
#include<stdio.h>
void main(){
  int seedNum,n1,n2,n3,n;
  printf("Enter the 4 digit seed number : ");
  scanf("%d",&seedNum);
  printf("Enter count to genrate random numbers : ");
  scanf("%d",&n);
  printf("Generated random numbers :\n%d\n",seedNum);
  for(int i=0; i<n ; i++){
    n1=seedNum*seedNum;
    n2=n1/100;
    n3=n2%10000;
    printf("%d\n",n3);
    seedNum=n3;
    }
}
```

OUTPUT: