OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH

TU DUBLIN

TECHNOLOGICAL UNIVERSITY DUBLIN

Assignment 2

Hypothesis Testing – Confidence Interval Program

Statistical analysis for engineers

By

Vimal Jaswal

B00122875

Master of Engineering in Internet of Things Technologies.

Damian Cox

School of Informatics & Engineering, Department of Engineering
TU Dublin – Blanchardstown Campus

Blanchardstown, Dublin 15

Declaration

I declare that the program for problem of this assignment have been written by me. I will be available for explaining anything written in the assignment which seems unclear.

Signature ___Vimal Jaswal____________________ Date ________30/8/2019____________

Summary

This assignment involves writing a computer program. It will generate a number of data values from a normal distribution with known parameters. The program will then calculate a confidence interval for the mean and variance. This step will be repeated a large number of times and the probability of the parameters being in their Intervals will be empirically
calculated.

Solution and Code written in C:

The program is given in appendix for testing. The Program can be run using online C language compiler such as the link given below, or you can use Code blocks IDE or any other IDE for test.

(https://www.onlinegdb.com/online_c_compiler)

The **rand()** function of C included in <math.h> is used for generating uniformly distributed random numbers.

The **srand(time(NULL))** function is also used in main() so that always different set of random numbers will be generated.

Box Muller Transform is being used to convert it to normally distributed random numbers.

“The Box-Muller transformations simply takes two variables that are uniformly distributed and sends them to two independent random variables with a standard normal distribution.”

$$Z_0 = R\cos(\Theta) = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

Reference:

Goodman, Jonathan. Lecture Notes on Monte Carlo Methods. Chapter 2: Simple Sampling of Gaussians Retrieved from https://www.math.nyu.edu/faculty/goodman/teaching/MonteCarlo2005/notes/GaussianSampling.pdf on March 16, 2018

You have to enter the value for n for generating random samples with its true mean and variance. The samples of n normally distributed numbers will then be used for calculating sample mean, sample variance and sample standard deviation.

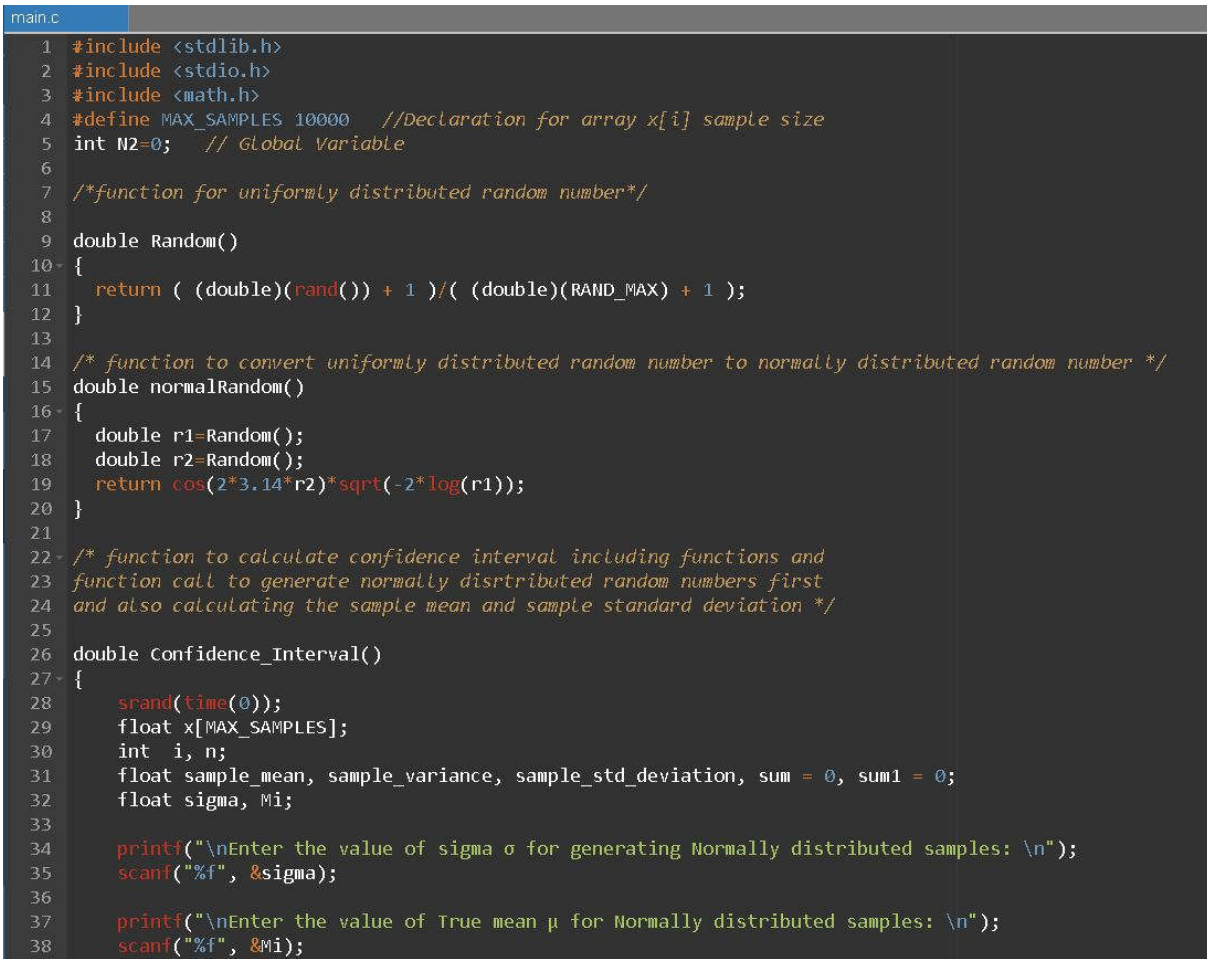
Further you have to choose a level of significance for generating confidence interval from the samples.

The program takes care of how many times the true mean value falls inside confidence interval counts for it and generate proportion.

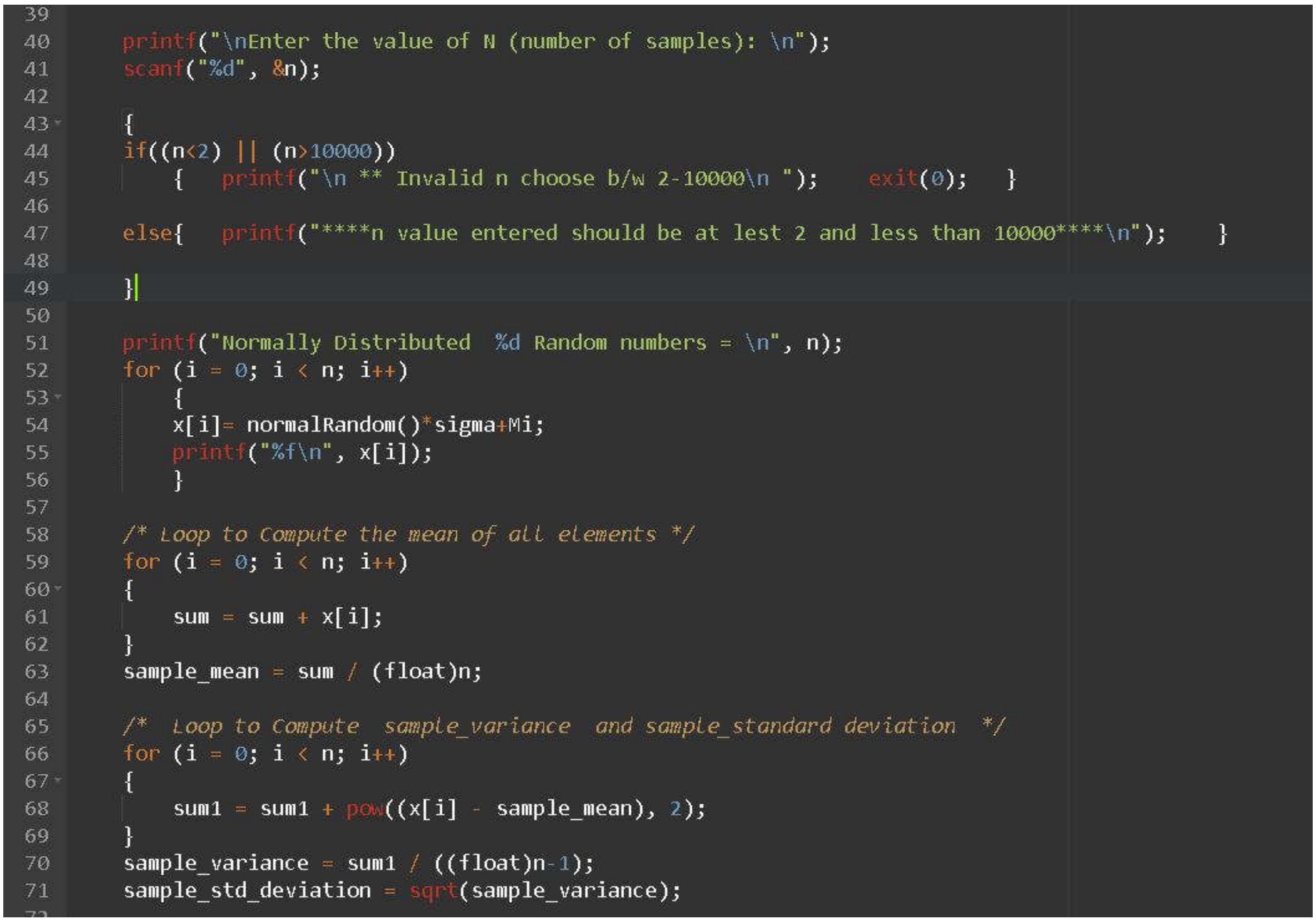
The Output for proportion depends upon number of samples. If sample values will be very less the true mean value may not fall inside the confidence interval. The reason is because T-test score should be used in that case for confidence interval, but I have written formula using z-tables score and its level of significance.

The Program works well with positive, negative and floating-point numbers. I have added some exception handling messages, but it still does not show error for some wrong inputs such as alphabetic character inputs during runtime. The program loop is set for 5 times repetition.

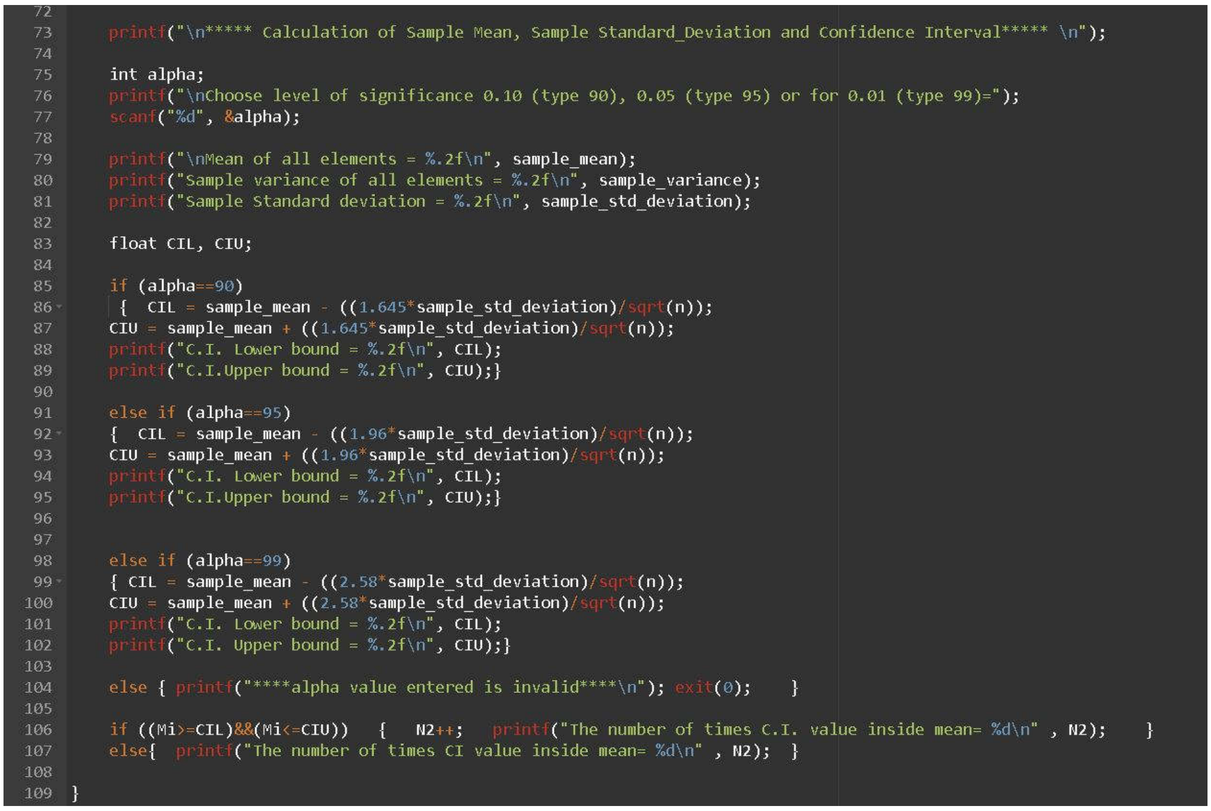
Below are the images for the program following output of the program as well.



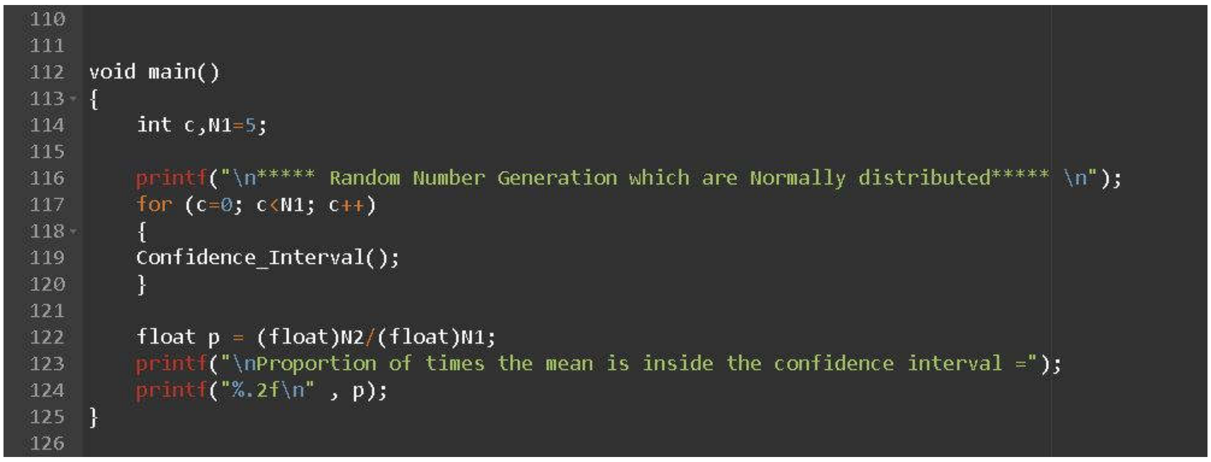
```c
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define MAX_SAMPLES 10000   //Declaration for array x[i] sample size
int N2=0;   // Global Variable

/*function for uniformly distributed random number*/

double Random()
{
  return ( (double)(rand()) + 1 )/( (double)(RAND_MAX) + 1 );
}

/* function to convert uniformly distributed random number to normally distributed random number */
double normalRandom()
{
  double r1=Random();
  double r2=Random();
  return cos(2*3.14*r2)*sqrt(-2*log(r1));
}

/* function to calculate confidence interval including functions and
function call to generate normally disrtributed random numbers first
and also calculating the sample mean and sample standard deviation */

double Confidence_Interval()
{
    srand(time(0));
    float x[MAX_SAMPLES];
    int  i, n;
    float sample_mean, sample_variance, sample_std_deviation, sum = 0, sum1 = 0;
    float sigma, Mi;

    printf("\nEnter the value of sigma σ for generating Normally distributed samples: \n");
    scanf("%f", &sigma);

    printf("\nEnter the value of True mean μ for Normally distributed samples: \n");
    scanf("%f", &Mi);
```



```c

    printf("\nEnter the value of N (number of samples): \n");
    scanf("%d", &n);

    {
    if((n<2) || (n>10000))
        {   printf("\n ** Invalid n choose b/w 2-10000\n ");    exit(0);    }

    else{   printf("****n value entered should be at lest 2 and less than 10000****\n");    }

    }

    printf("Normally Distributed  %d Random numbers = \n", n);
    for (i = 0; i < n; i++)
        {
        x[i]= normalRandom()*sigma+Mi;
        printf("%f\n", x[i]);
        }

    /* Loop to Compute the mean of all elements */
    for (i = 0; i < n; i++)
    {
        sum = sum + x[i];
    }
    sample_mean = sum / (float)n;

    /*  Loop to Compute  sample_variance  and sample_standard deviation  */
    for (i = 0; i < n; i++)
    {
        sum1 = sum1 + pow((x[i] - sample_mean), 2);
    }
    sample_variance = sum1 / ((float)n-1);
    sample_std_deviation = sqrt(sample_variance);
```

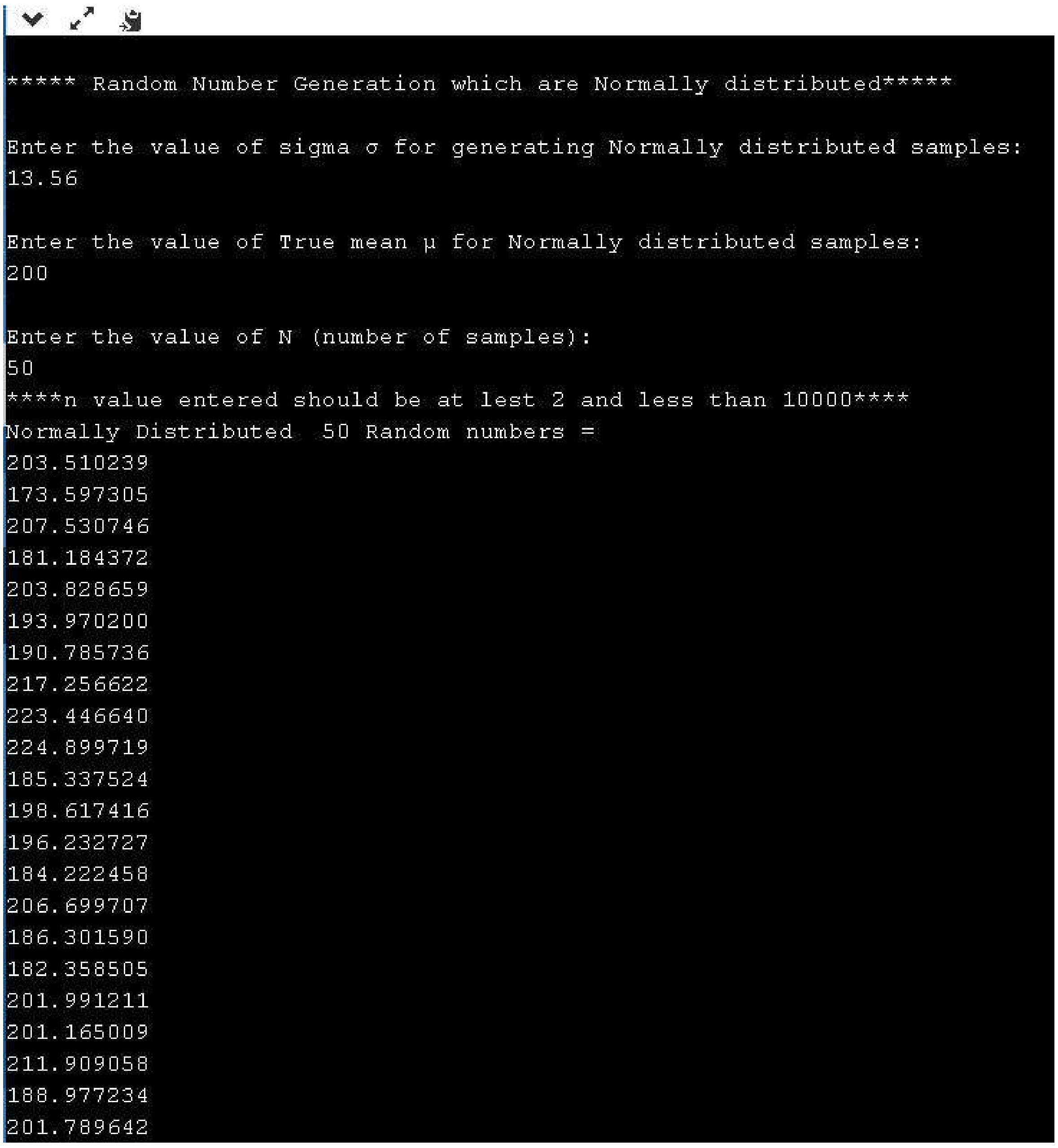


```c

    printf("\n***** Calculation of Sample Mean, Sample Standard_Deviation and Confidence Interval***** \n");

    int alpha;
    printf("\nChoose level of significance 0.10 (type 90), 0.05 (type 95) or for 0.01 (type 99)=");
    scanf("%d", &alpha);

    printf("\nMean of all elements = %.2f\n", sample_mean);
    printf("Sample variance of all elements = %.2f\n", sample_variance);
    printf("Sample Standard deviation = %.2f\n", sample_std_deviation);

    float CIL, CIU;

    if (alpha==90)
     {  CIL = sample_mean - ((1.645*sample_std_deviation)/sqrt(n));
    CIU = sample_mean + ((1.645*sample_std_deviation)/sqrt(n));
    printf("C.I. Lower bound = %.2f\n", CIL);
    printf("C.I.Upper bound = %.2f\n", CIU);}

    else if (alpha==95)
    {  CIL = sample_mean - ((1.96*sample_std_deviation)/sqrt(n));
    CIU = sample_mean + ((1.96*sample_std_deviation)/sqrt(n));
    printf("C.I. Lower bound = %.2f\n", CIL);
    printf("C.I.Upper bound = %.2f\n", CIU);}


    else if (alpha==99)
    { CIL = sample_mean - ((2.58*sample_std_deviation)/sqrt(n));
    CIU = sample_mean + ((2.58*sample_std_deviation)/sqrt(n));
    printf("C.I. Lower bound = %.2f\n", CIL);
    printf("C.I. Upper bound = %.2f\n", CIU);}

    else { printf("****alpha value entered is invalid****\n"); exit(0);    }

    if ((Mi>=CIL)&&(Mi<=CIU))   {   N2++;   printf("The number of times C.I. value inside mean= %d\n" , N2);    }
    else{  printf("The number of times CI value inside mean= %d\n" , N2);  }

}
```

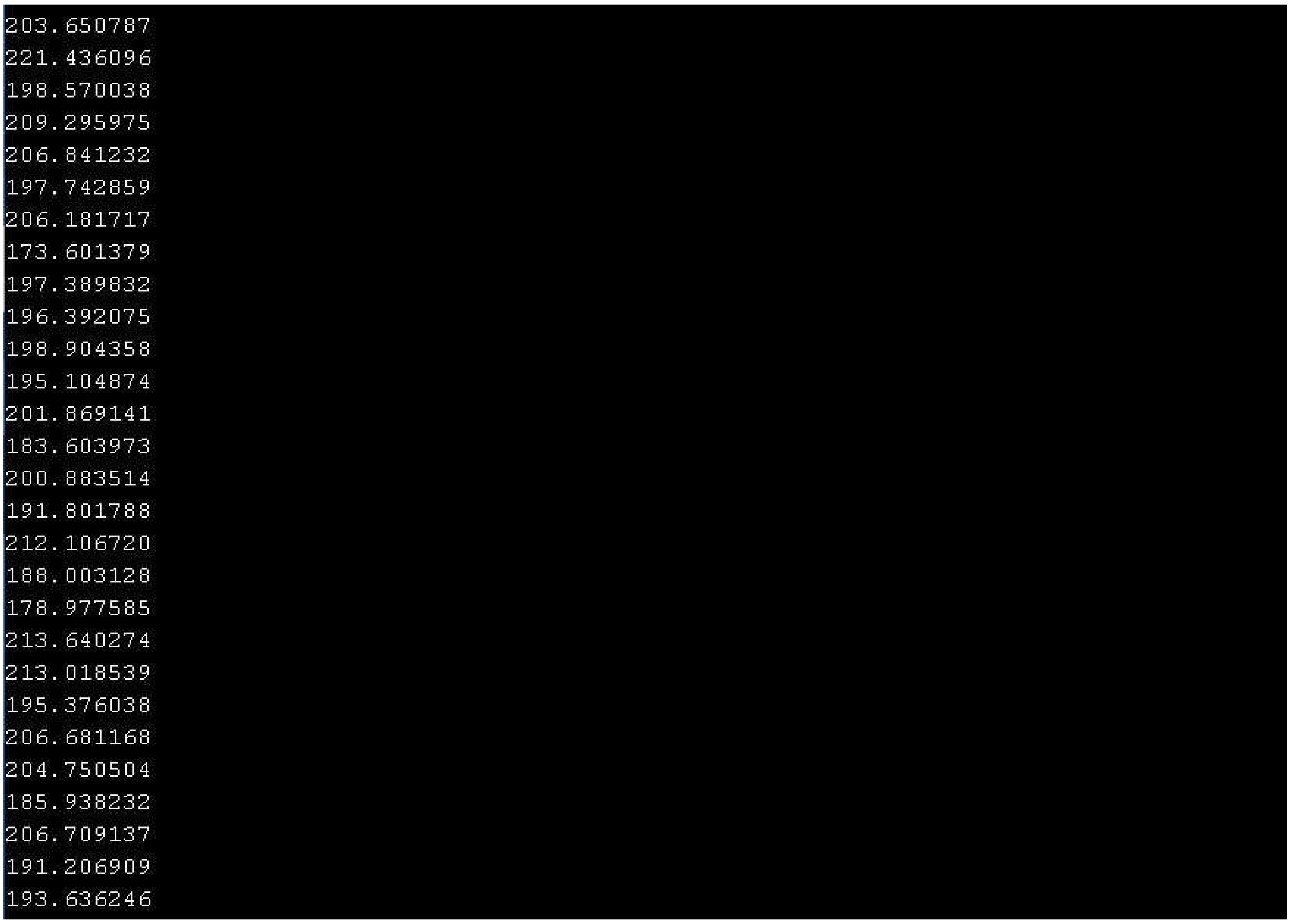


```c


void main()
{
    int c,N1=5;

    printf("\n***** Random Number Generation which are Normally distributed***** \n");
    for (c=0; c<N1; c++)
    {
    Confidence_Interval();
    }

    float p = (float)N2/(float)N1;
    printf("\nProportion of times the mean is inside the confidence interval =");
    printf("%.2f\n" , p);
}

```

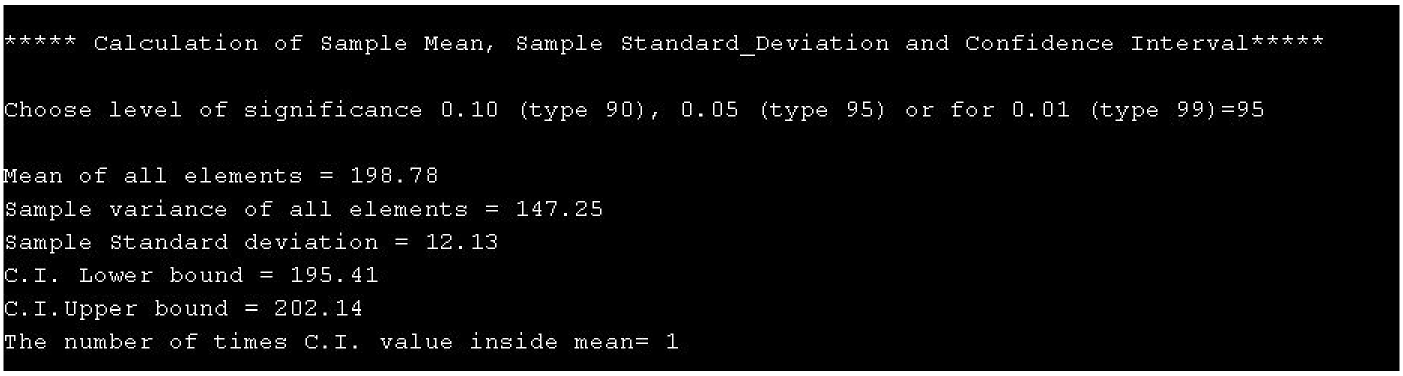
Output as per my input data:



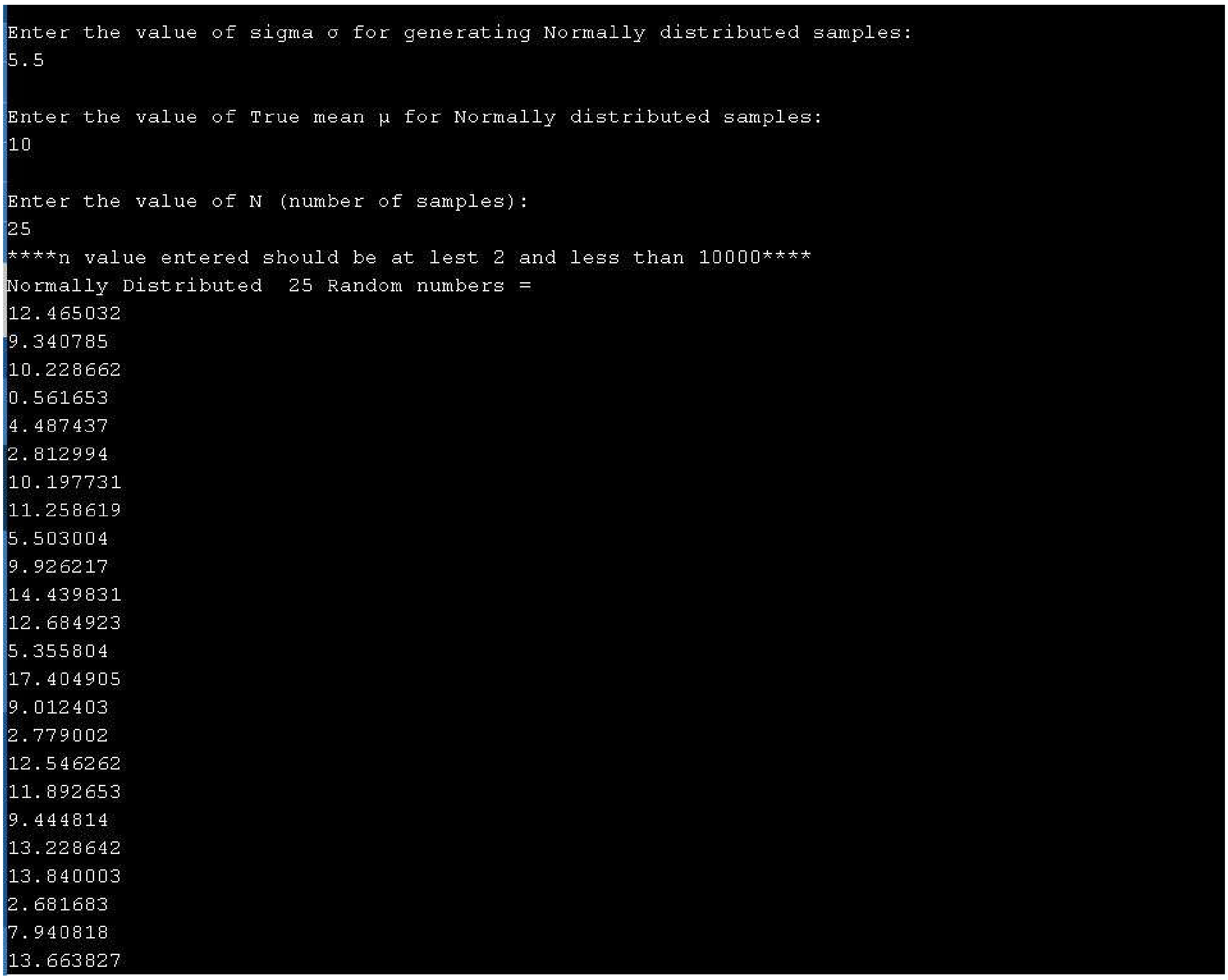
```
***** Random Number Generation which are Normally distributed*****

Enter the value of sigma σ for generating Normally distributed samples:
13.56

Enter the value of True mean μ for Normally distributed samples:
200

Enter the value of N (number of samples):
50
****n value entered should be at lest 2 and less than 10000****
Normally Distributed  50 Random numbers =
203.510239
173.597305
207.530746
181.184372
203.828659
193.970200
190.785736
217.256622
223.446640
224.899719
185.337524
198.617416
196.232727
184.222458
206.699707
186.301590
182.358505
201.991211
201.165009
211.909058
188.977234
201.789642
```



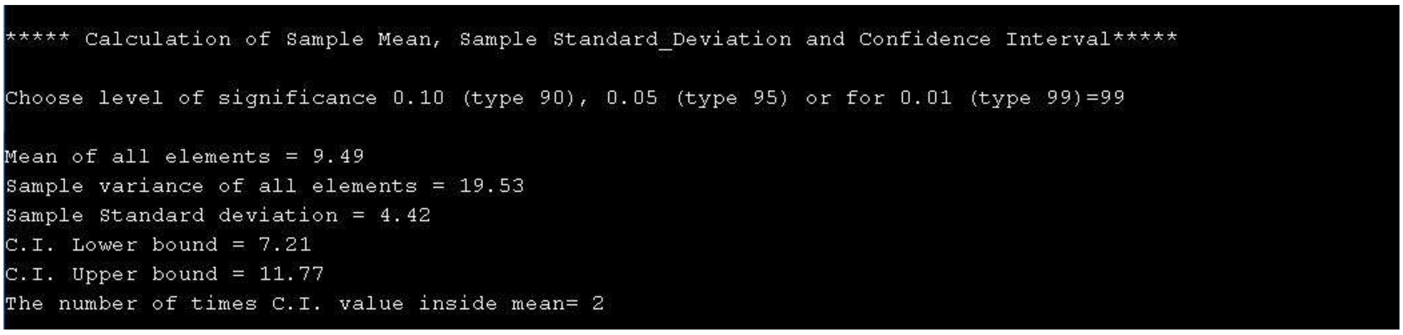
```
203.650787
221.436096
198.570038
209.295975
206.841232
197.742859
206.181717
173.601379
197.389832
196.392075
198.904358
195.104874
201.869141
183.603973
200.883514
191.801788
212.106720
188.003128
178.977585
213.640274
213.018539
195.376038
206.681168
204.750504
185.938232
206.709137
191.206909
193.636246
```



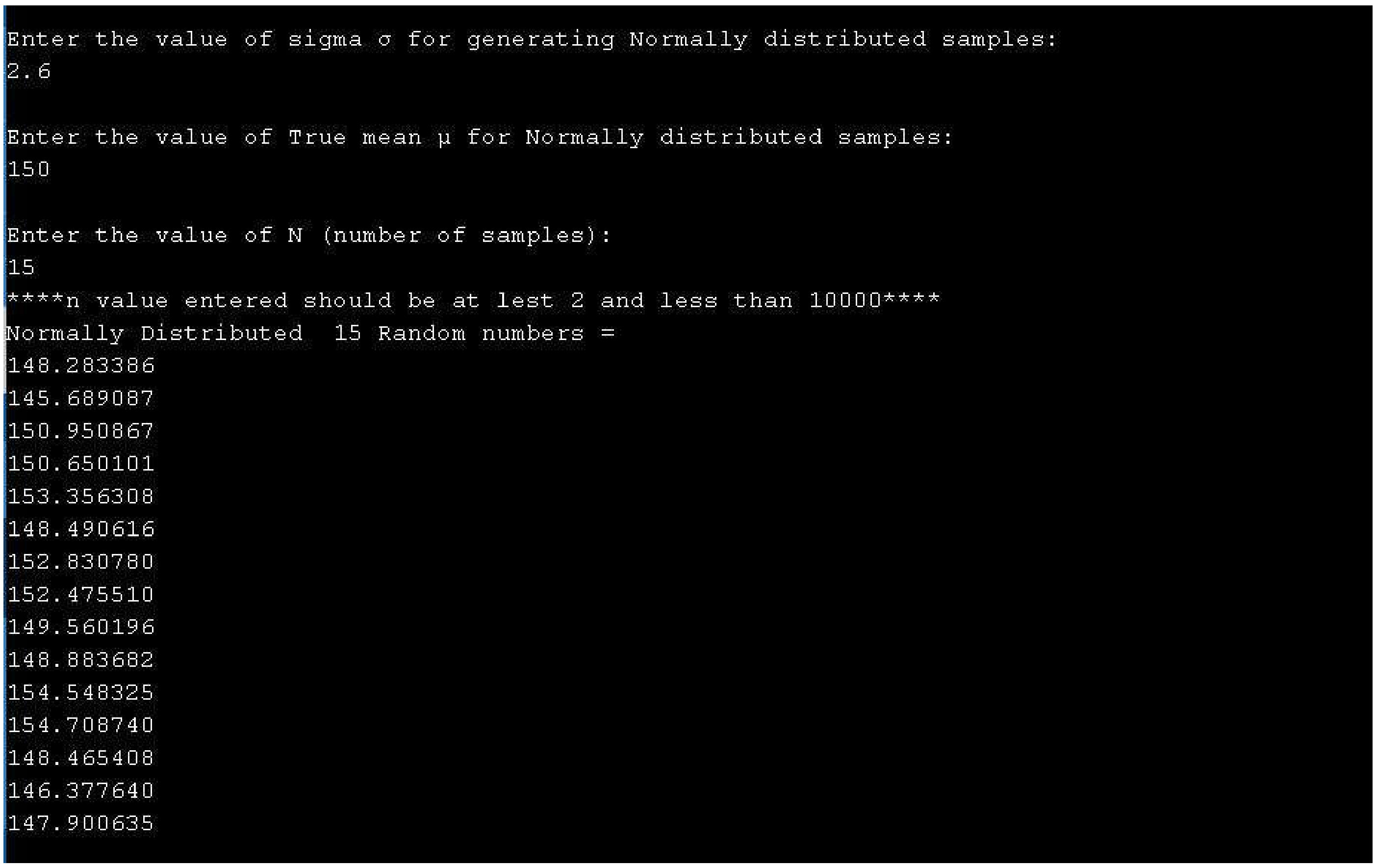
```
***** Calculation of Sample Mean, Sample Standard_Deviation and Confidence Interval*****

Choose level of significance 0.10 (type 90), 0.05 (type 95) or for 0.01 (type 99)=95

Mean of all elements = 198.78
Sample variance of all elements = 147.25
Sample Standard deviation = 12.13
C.I. Lower bound = 195.41
C.I.Upper bound = 202.14
The number of times C.I. value inside mean= 1
```



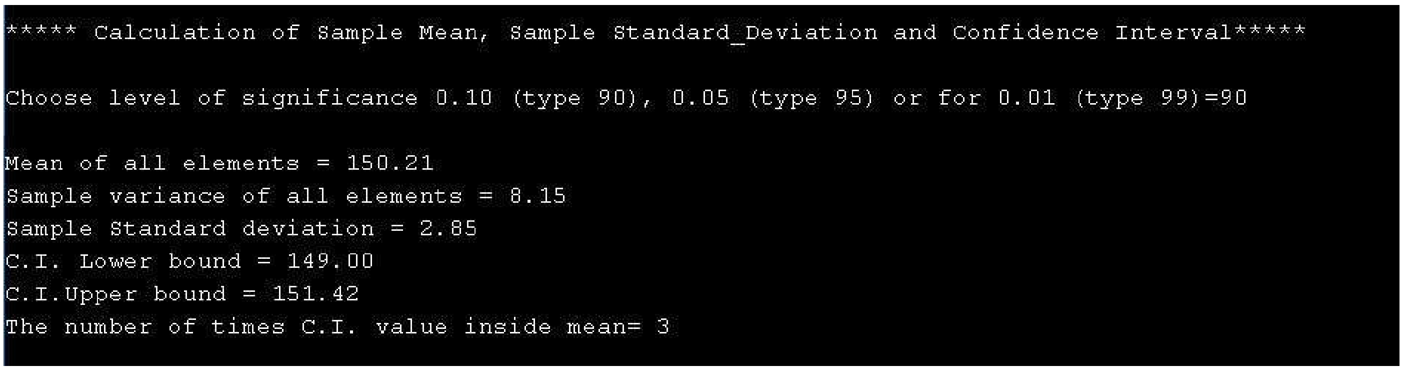
```
Enter the value of sigma σ for generating Normally distributed samples:
5.5

Enter the value of True mean μ for Normally distributed samples:
10

Enter the value of N (number of samples):
25
****n value entered should be at lest 2 and less than 10000****
Normally Distributed  25 Random numbers =
12.465032
9.340785
10.228662
0.561653
4.487437
2.812994
10.197731
11.258619
5.503004
9.926217
14.439831
12.684923
5.355804
17.404905
9.012403
2.779002
12.546262
11.892653
9.444814
13.228642
13.840003
2.681683
7.940818
13.663827
```



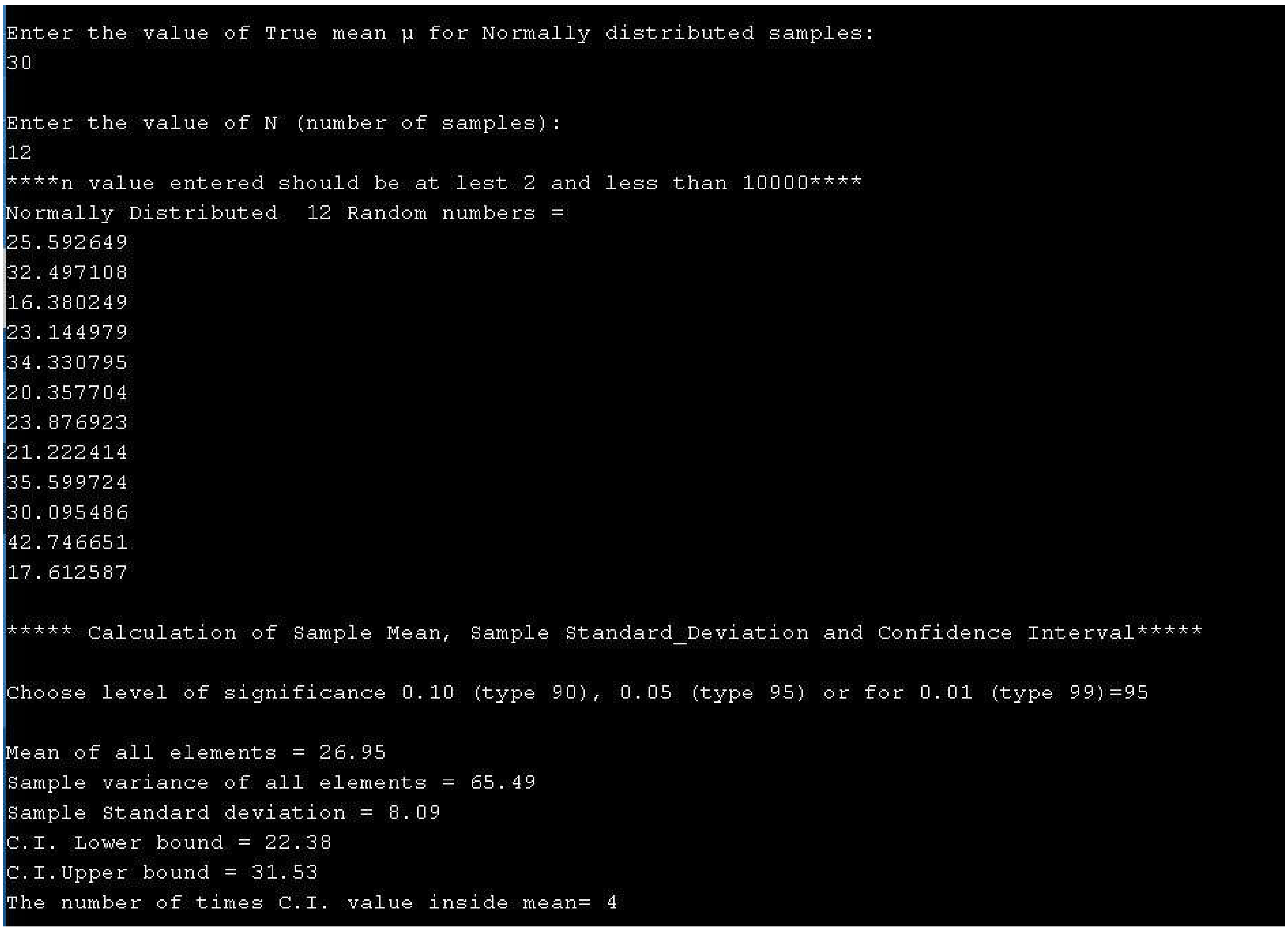
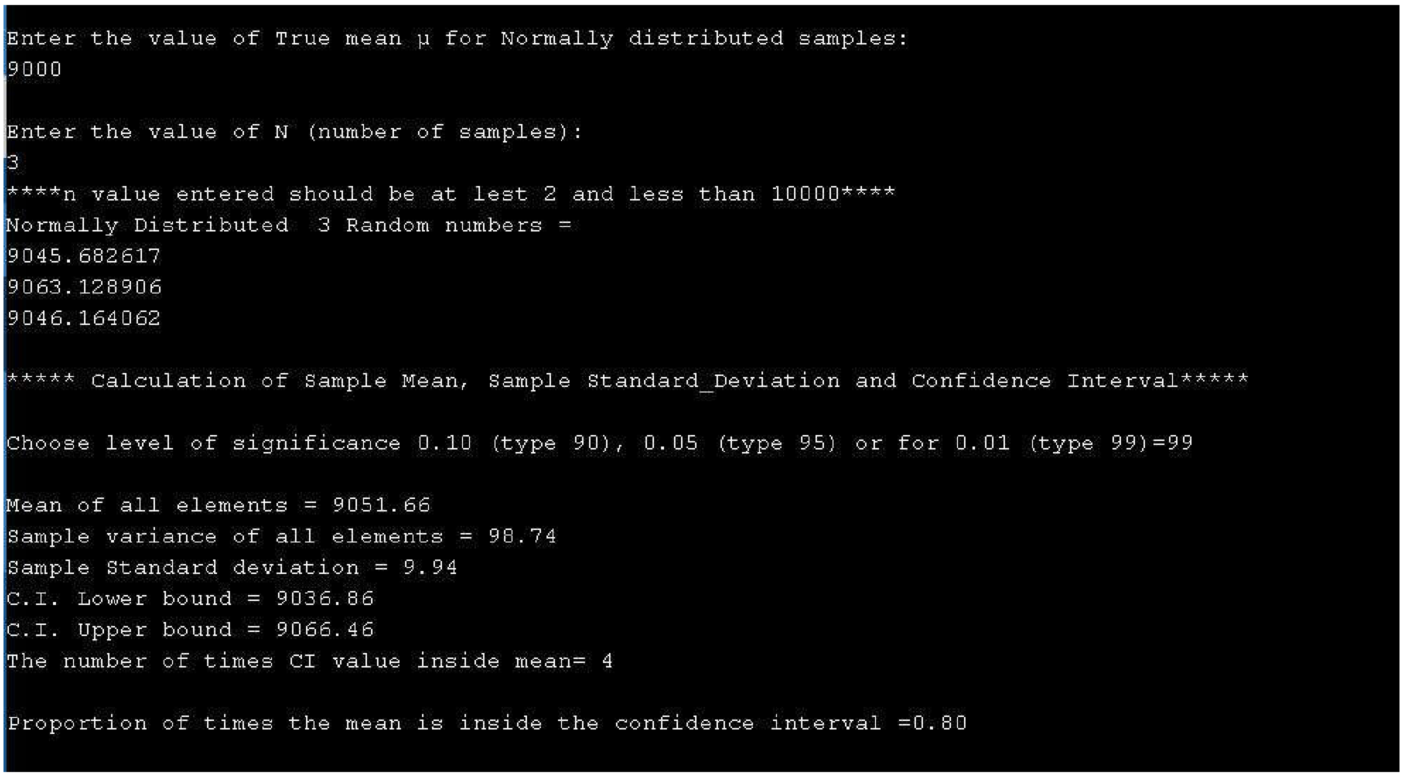
```
***** Calculation of Sample Mean, Sample Standard_Deviation and Confidence Interval*****

Choose level of significance 0.10 (type 90), 0.05 (type 95) or for 0.01 (type 99)=99

Mean of all elements = 9.49
Sample variance of all elements = 19.53
Sample Standard deviation = 4.42
C.I. Lower bound = 7.21
C.I. Upper bound = 11.77
The number of times C.I. value inside mean= 2
```



```
Enter the value of sigma σ for generating Normally distributed samples:
2.6

Enter the value of True mean μ for Normally distributed samples:
150

Enter the value of N (number of samples):
15
****n value entered should be at lest 2 and less than 10000****
Normally Distributed  15 Random numbers =
148.283386
145.689087
150.950867
150.650101
153.356308
148.490616
152.830780
152.475510
149.560196
148.883682
154.548325
154.708740
148.465408
146.377640
147.900635
```



```
***** Calculation of Sample Mean, Sample Standard_Deviation and Confidence Interval*****

Choose level of significance 0.10 (type 90), 0.05 (type 95) or for 0.01 (type 99)=90

Mean of all elements = 150.21
Sample variance of all elements = 8.15
Sample Standard deviation = 2.85
C.I. Lower bound = 149.00
C.I.Upper bound = 151.42
The number of times C.I. value inside mean= 3
```


```
Enter the value of True mean µ for Normally distributed samples:
30

Enter the value of N (number of samples):
12
****n value entered should be at lest 2 and less than 10000****
Normally Distributed  12 Random numbers =
25.592649
32.497108
16.380249
23.144979
34.330795
20.357704
23.876923
21.222414
35.599724
30.095486
42.746651
17.612587

***** Calculation of Sample Mean, Sample Standard_Deviation and Confidence Interval*****

Choose level of significance 0.10 (type 90), 0.05 (type 95) or for 0.01 (type 99)=95

Mean of all elements = 26.95
Sample variance of all elements = 65.49
Sample Standard deviation = 8.09
C.I. Lower bound = 22.38
C.I.Upper bound = 31.53
The number of times C.I. value inside mean= 4
```

```
Enter the value of True mean µ for Normally distributed samples:
9000

Enter the value of N (number of samples):
3
****n value entered should be at lest 2 and less than 10000****
Normally Distributed  3 Random numbers =
9045.682617
9063.128906
9046.164062

***** Calculation of Sample Mean, Sample Standard_Deviation and Confidence Interval*****

Choose level of significance 0.10 (type 90), 0.05 (type 95) or for 0.01 (type 99)=99

Mean of all elements = 9051.66
Sample variance of all elements = 98.74
Sample Standard deviation = 9.94
C.I. Lower bound = 9036.86
C.I. Upper bound = 9066.46
The number of times CI value inside mean= 4

Proportion of times the mean is inside the confidence interval =0.80
```

APPENDIX

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define MAX_SAMPLES 10000   //Declaration for array x[i] sample size
int N2=0;   // Global Variable

/*function for uniformly distributed random number*/

double Random()
{
  return ( (double)(rand()) + 1 )/( (double)(RAND_MAX) + 1 );
}

/* function to convert uniformly distributed random number to normally distributed random number */
double normalRandom()
{
  double r1=Random();
  double r2=Random();
  return cos(2*3.14*r2)*sqrt(-2*log(r1));
}

/* function to calculate confidence interval including functions and
function call to generate normally disrtributed random numbers first
and also calculating the sample mean and sample standard deviation */

double Confidence_Interval()
{
   srand(time(0));
   float x[MAX_SAMPLES];
   int  i, n;
   float sample_mean, sample_variance, sample_std_deviation, sum = 0, sum1 = 0;
   float sigma, Mi;

   printf("\nEnter the value of sigma σ for generating Normally distributed samples: \n");
   scanf("%f", &sigma);

   printf("\nEnter the value of True mean μ for Normally distributed samples: \n");
   scanf("%f", &Mi);

   printf("\nEnter the value of N (number of samples): \n");
   scanf("%d", &n);

   {
   if((n<2) ||( n>10000))
     {   printf("\n ** Invalid n choose b/w 2-10000\n ");    exit(0);   }

   else{   printf("****n value entered should be at lest 2 and less than 10000****\n");    }

   }

   printf("Normally Distributed  %d Random numbers = \n", n);
   for (i = 0; i < n; i++)
     {
     x[i]= normalRandom()*sigma+Mi;
```

```
      printf("%f\n", x[i]);
      }

   /* Loop to Compute the mean of all elements */
   for (i = 0; i < n; i++)
   {
      sum = sum + x[i];
   }
   sample_mean = sum / (float)n;

   /*  Loop to Compute  sample_variance  and sample_standard deviation  */
   for (i = 0; i < n; i++)
   {
      sum1 = sum1 + pow((x[i] - sample_mean), 2);
   }
   sample_variance = sum1 / ((float)n-1);
   sample_std_deviation = sqrt(sample_variance);

   printf("\n***** Calculation of Sample Mean, Sample Standard_Deviation and Confidence Interval***** \n");

   int alpha;
   printf("\nChoose level of significance 0.10 (type 90), 0.05 (type 95) or for 0.01 (type 99)=");
   scanf("%d", &alpha);

   printf("\nMean of all elements = %.2f\n", sample_mean);
   printf("Sample variance of all elements = %.2f\n", sample_variance);
   printf("Sample Standard deviation = %.2f\n", sample_std_deviation);

   float CIL, CIU;

   if (alpha==90)
    {  CIL = sample_mean - ((1.645*sample_std_deviation)/sqrt(n));
   CIU = sample_mean + ((1.645*sample_std_deviation)/sqrt(n));
   printf("C.I. Lower bound = %.2f\n", CIL);
   printf("C.I.Upper bound = %.2f\n", CIU);}

   else if (alpha==95)
   {  CIL = sample_mean - ((1.96*sample_std_deviation)/sqrt(n));
   CIU = sample_mean + ((1.96*sample_std_deviation)/sqrt(n));
   printf("C.I. Lower bound = %.2f\n", CIL);
   printf("C.I.Upper bound = %.2f\n", CIU);}


   else if (alpha==99)
   { CIL = sample_mean - ((2.58*sample_std_deviation)/sqrt(n));
   CIU = sample_mean + ((2.58*sample_std_deviation)/sqrt(n));
   printf("C.I. Lower bound = %.2f\n", CIL);
   printf("C.I. Upper bound = %.2f\n", CIU);}

   else { printf("****alpha value entered is invalid****\n"); exit(0);    }

   if ((Mi>=CIL)&&(Mi<=CIU))     {   N2++;   printf("The number of times C.I. value inside mean= %d\n" , N2);    }
   else{  printf("The number of times CI value inside mean= %d\n" , N2);  }

}
```

```
void main()
{
   int c,N1=5;

   printf("\n***** Random Number Generation which are Normally distributed***** \n");
   for (c=0; c<N1; c++)
   {
   Confidence_Interval();
   }

   float p = (float)N2/(float)N1;
   printf("\nProportion of times the mean is inside the confidence interval =");
   printf("%.2f\n" , p);
}
```