

**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.”



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.

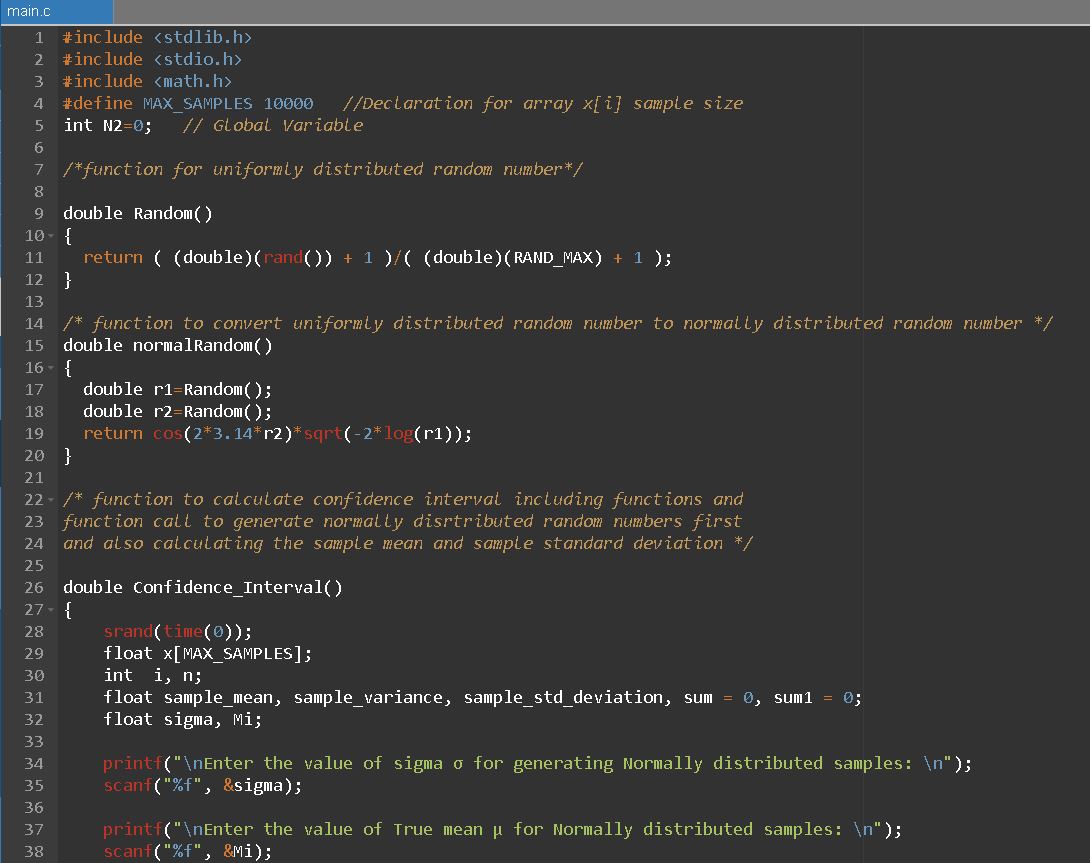
Further you have to choose a level of significance for generating confidence interval from the samples.

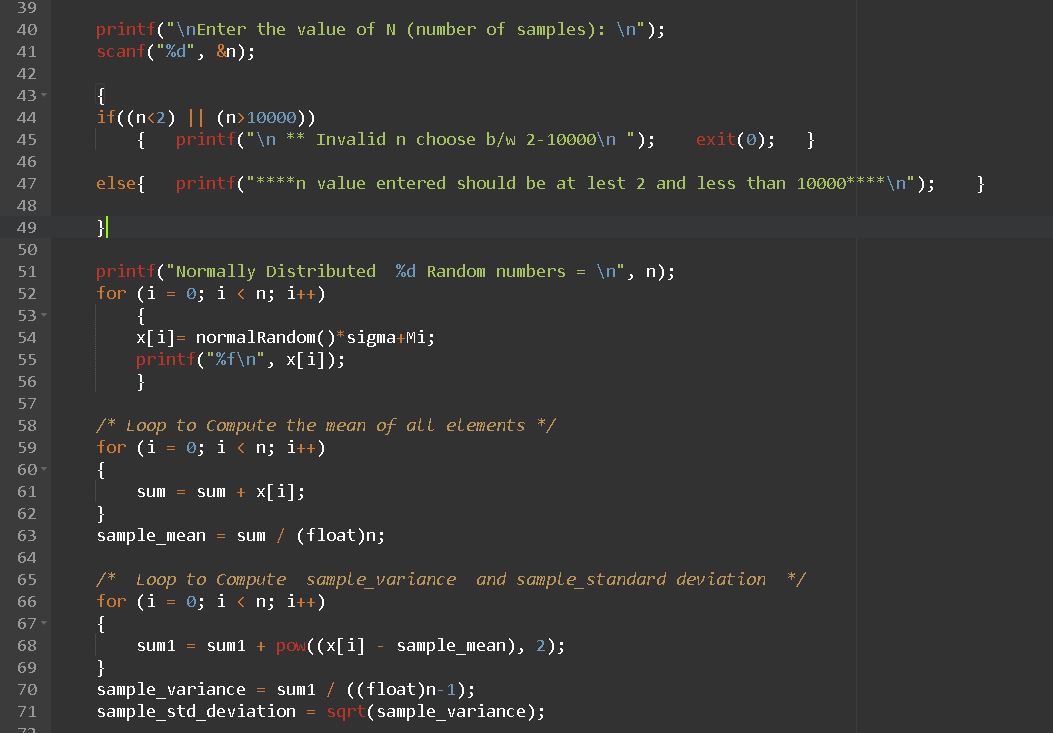
The program takes care of how many times the true mean value falls inside confidence interval counts for it and generate proportion.

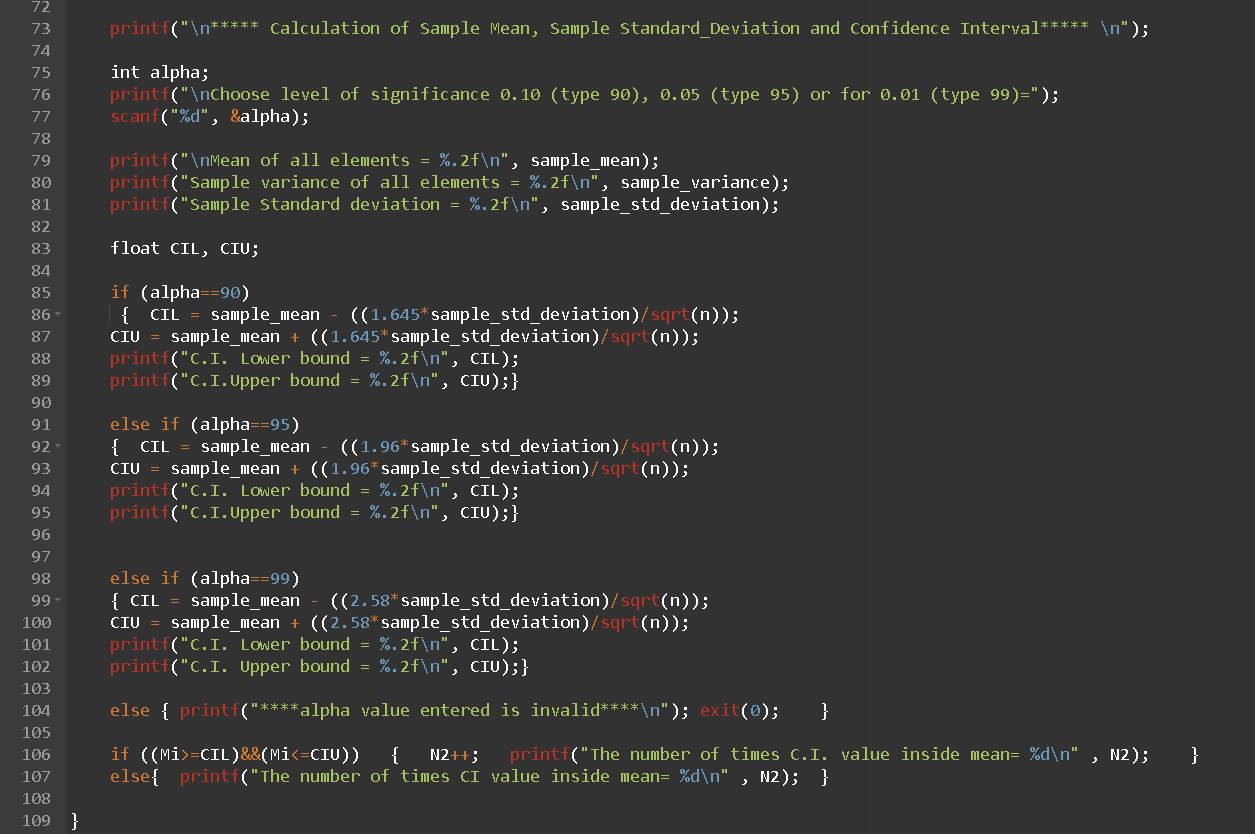
The Output for proportion depends upon number of samples. If sample values will be very less the true mean value may not fall inside the confidence interval. The reason is because T-test score should be used in that case for confidence interval, but I have written formula using z-tables score and its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

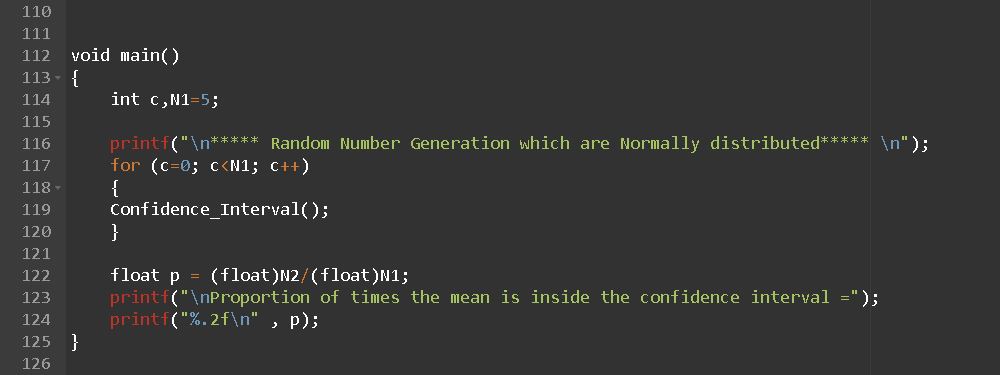
The Program works well with positive, negative and floating-point numbers. I have added some exception handling messages, but it still does not show error for some wrong inputs such as alphabetic character inputs during runtime. The program loop is set for 5 times repetition.

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

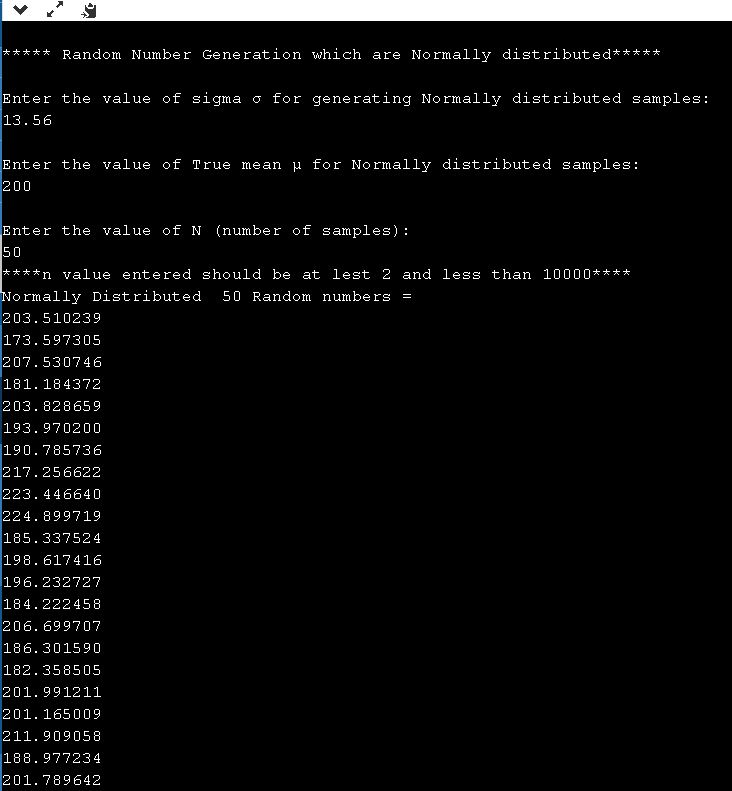


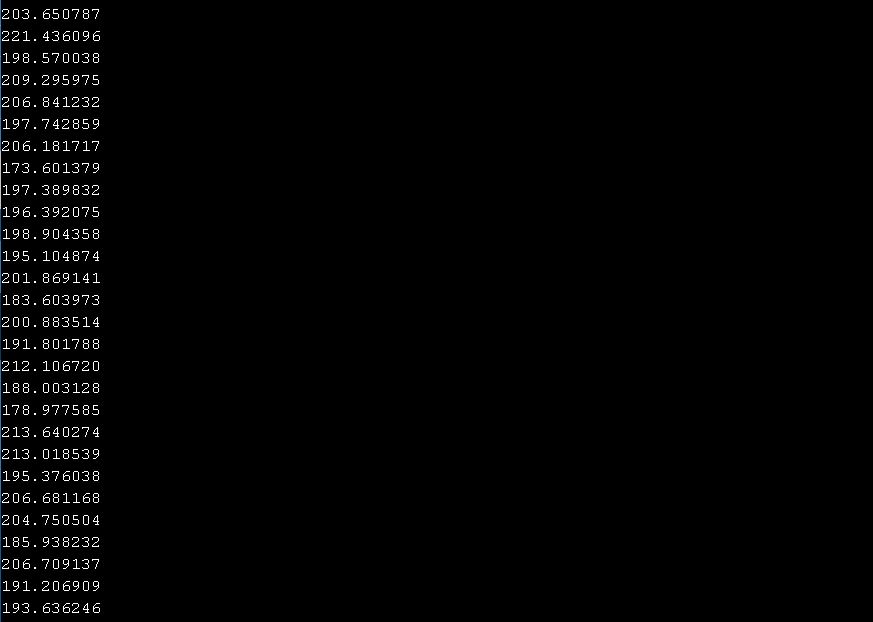


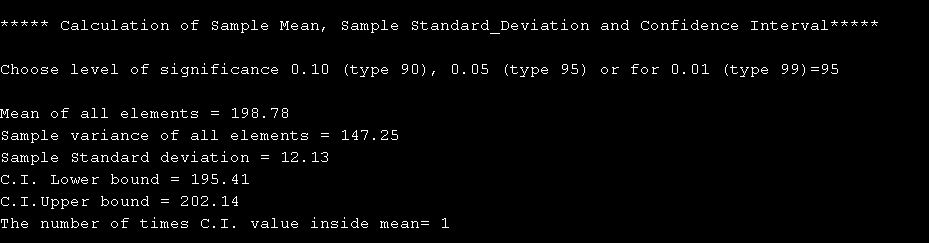


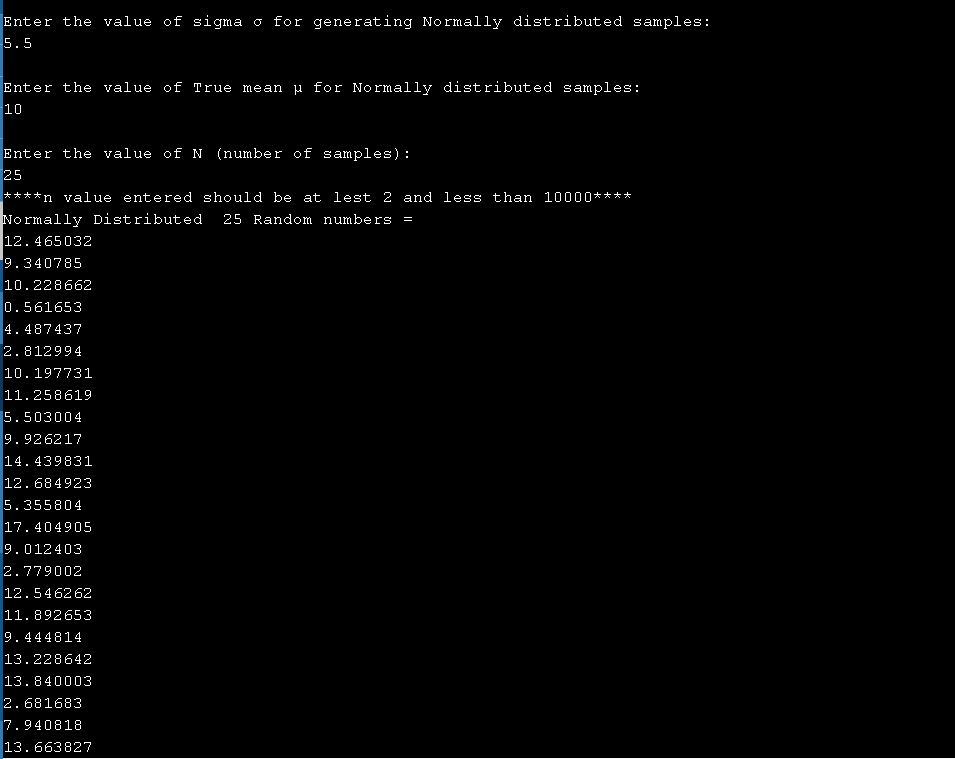


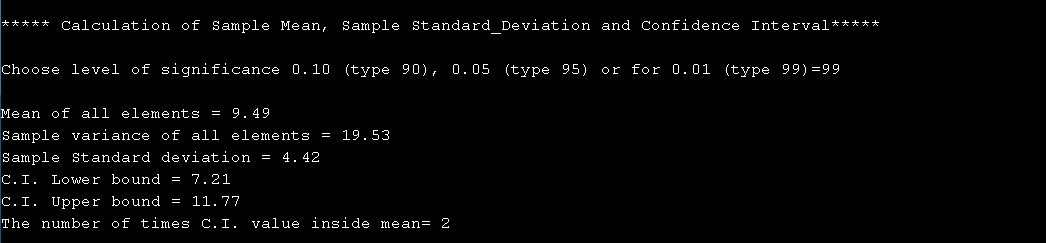
**Output as per my input data:**



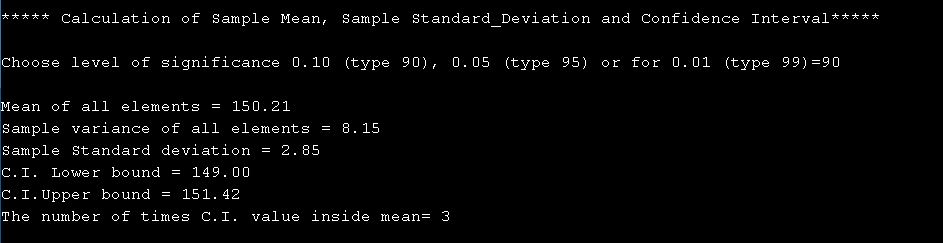


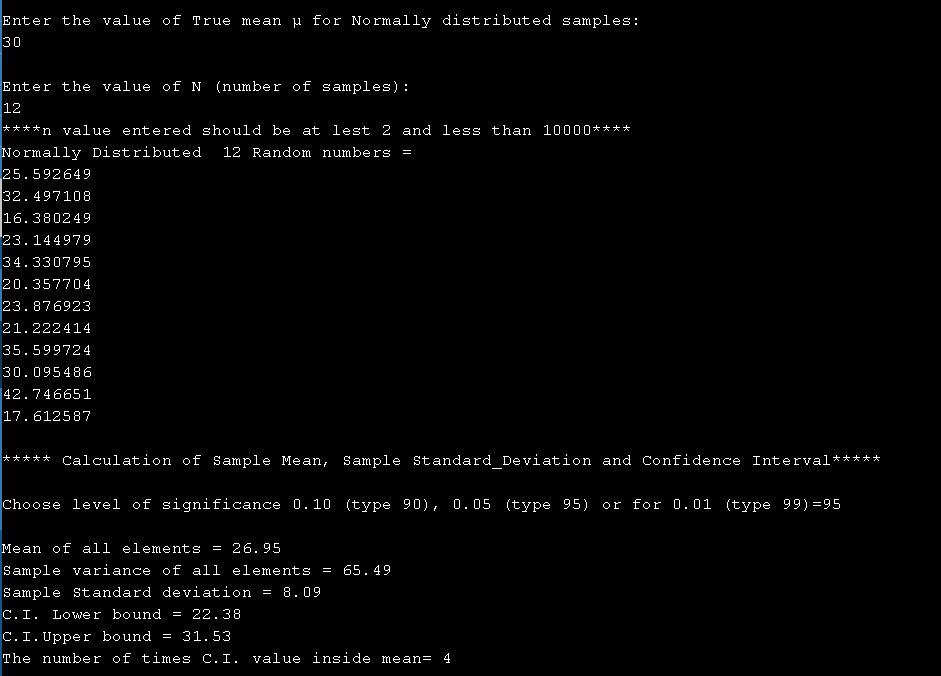


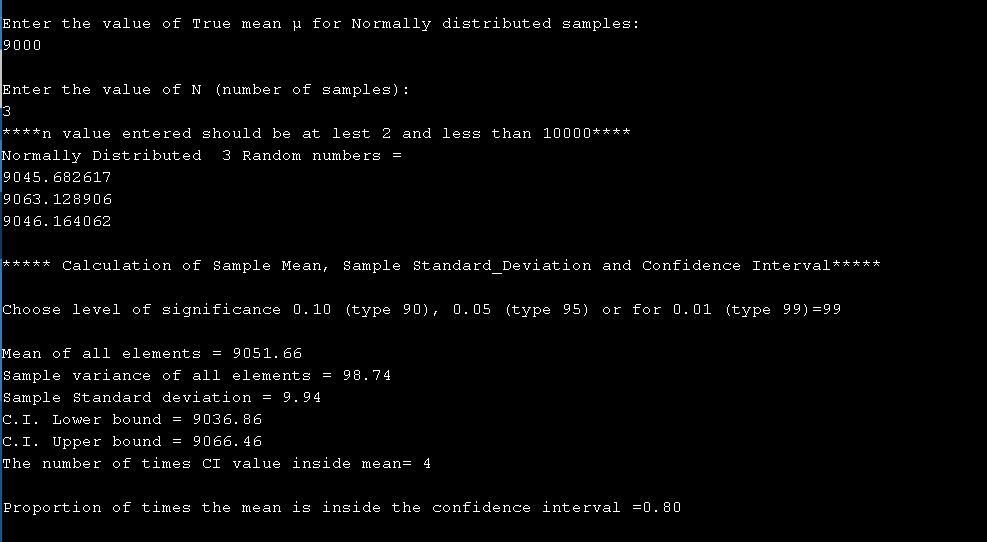












**APPENDIX**

**#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);**

**}**