Lab 1: Write a C program to find the growth in national consumption for five years using Distributed Lag Model given below:

I = 2 + 0.1 Y-1

Y = 45.45 + 2.27 (I +G)

T = 0.2 Y

C = 20 + 0.7 (Y – T)

Assume the initial value of Y-1 is 80 and take the governmental expenditure in the 5 years to be as follows:

Year	G
1	20
2	25
3	30
4	35
5	40

Lab-1 **Solution in C**

```
#include<stdio.h>

#include<conio.h>

int main()

{

    float Y_1,Y,I,T,C[5];

    float G[5]={20,25,30,35,40};

    int i;

    printf("Enter initial value of lagged variable Y_1:");

    scanf("%f",&Y_1);

    printf("\nThe growth in consumption is given following

    tables:\n"); printf("\nYear \t \t Consumption\n");

    for(i=0;i<5;i++){
```

```
            I=2+0.1*Y_1;
            Y=45.45+2.27*(I+G[i]);
            T=0.2*Y;
            C[i]=20+0.7*(Y-T);
            printf("\n %d \t \t %.2f",i+1,C[i]);
            Y_1=Y;
        }
        getch();
        return 0;
}
```

OUTPUT

```
C:\Users\Aayush\Desktop\MB

Enter initial value of lagged variable Y_1:80

The growth in consumption is given following tables:

Year                Consumption

 1                  83.59
 2                  94.21
 3                  102.98
 4                  111.32
 5                  119.57
--------------------------------
Process exited after 3.024 seconds with return value 0
Press any key to continue . . .
```

Lab 2: Customers arrive in a bank according to a Poisson's process with a mean inter arrival time of 10 minutes. Customers spend an average of 5 minutes on the single available counter, and leave.

Write a program in C to find:
I. Probability that a customer will not have to wait at the counter.
II. Expected number of customers in the bank.
III. Time can a customer expect to spend in the bank.

<u>Lab-2 Solution:</u>

```
#include<stdio.h>
int main() {
   float inter_arrival_time, avg_service_time, lambda, mu, rho;
   float p_no_wait, expected_customers, expected_time;

   printf("Enter inter-arrival time of customers (minutes): ");
   scanf("%f", &inter_arrival_time);
   printf("Enter average service time of customers (minutes): ");
   scanf("%f", &avg_service_time);

   lambda = 60 / inter_arrival_time;
   mu = 60 / avg_service_time;

   if (lambda >= mu) {
      printf("\nThe system is unstable (arrival rate exceeds service rate).\n");
      return 0;
   }
```

```c
    rho = lambda / mu;
    p_no_wait = 1 - rho;
    expected_customers = lambda / (mu - lambda);
    expected_time = (1 / (mu - lambda)) * 60;

    printf("\nProbability that a customer will not have to wait at the counter: %.2f",
p_no_wait);
    printf("\nExpected number of customers in the bank: %.2f", expected_customers);
    printf("\nExpected time a customer spends in the bank: %.2f minutes\n",
expected_time);

    return 0;
}
```

OUTPUT

```
C:\Users\Aayush\Desktop\ME
Enter inter-arrival time of customers (minutes): 10
Enter average service time of customers (minutes): 5

Probability that a customer will not have to wait at the counter: 0.50
Expected number of customers in the bank: 1.00
Expected time a customer spends in the bank: 10.00 minutes

--------------------------------
Process exited after 7.894 seconds with return value 0
Press any key to continue . . .
```

Lab 3: At the ticket counter of the football stadium, people come in queue and purchase tickets. Arrival rate of customers is 1/min. It takes an average of 20 seconds to purchase the ticket.

WAP in C to calculate total time spent by a sports fan to be seated in his seat, if it takes 1.5 minutes to reach the correct seat after purchasing the ticket. If a fan comes exactly 2 minutes before the game starts, can a sports fan expect to be seated for the kick-off?

Lab-3 Solutions:
Given Information:
1. Arrival Rate: 1/min (customers arrive at the rate of 1 per minute).
2. Service Rate: Since it takes 20 seconds to serve a customer, we convert it to a rate as 60/20 = 3 customers/minute.
3. Travel time to the seat: 1.5 minutes.
4. Time before the game starts: 2 minutes

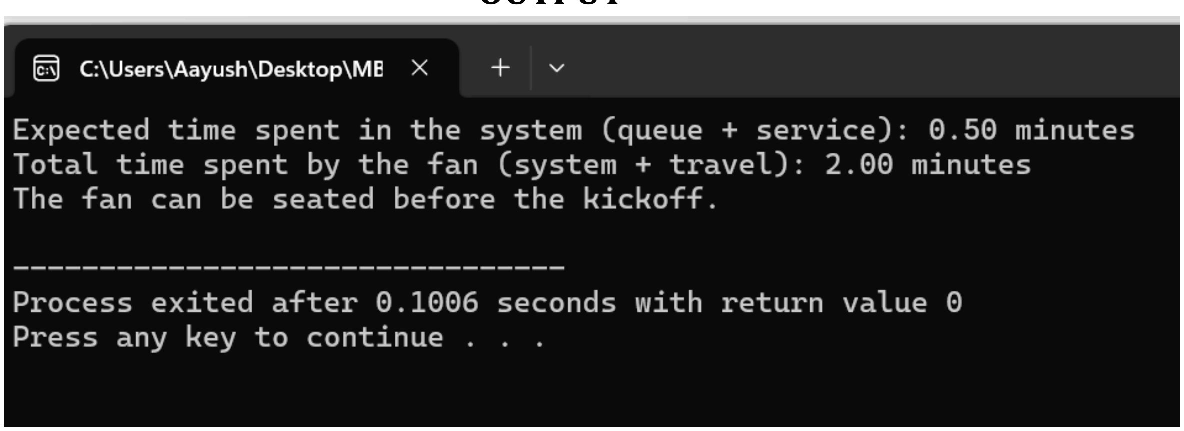
We need to calculate:

1. The total time a fan spends in the system (waiting in the queue, purchasing a ticket, and walking to the seat).
2. Check if the fan can be seated before the kickoff.

```
#include <stdio.h>

int main() {
    float arrival_rate = 1.0;    // Customers per minute
    float service_rate = 3.0;   // Customers per minute (since 20 sec = 3/min)
    float travel_time = 1.5;    // Time to reach the seat (in minutes)
    float time_before_game = 2.0; // Time before game starts (in minutes)


    float utilization = arrival_rate / service_rate;


    if (utilization >= 1) {
        printf("The system is unstable. Arrival rate exceeds service rate.\n");
```

```
    return 0;
  }


  float time_in_system = 1 / (service_rate - arrival_rate);


  float total_time = time_in_system + travel_time;


  printf("Expected time spent in the system (queue + service): %.2f minutes\n", time_in_system);
  printf("Total time spent by the fan (system + travel): %.2f minutes\n", total_time);


  if (total_time <= time_before_game) {
    printf("The fan can be seated before the kickoff.\n");
  } else {
    printf("The fan cannot be seated before the kickoff.\n");
  }

  return 0;
}
```

OUTPUT



Lab 4: In a single pump service station, vehicles arrive for fuelling with an average of 5 minutes between arrivals. If an hour is taken as a unit of time, cars arrive according to Poison's process with an average of λ= 12 cars/hr.
Write a C program to generate Poisson distribution for x = 0,1,2, ... 15.

<u>Lab-4 Solution:</u>

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

// Function to calculate factorial
float factorial(int n) {
   float fact = 1,i;
   for ( i = 1; i <= n; i++) {
      fact *= i;
   }
   return fact;
}

int main() {
   float lambda = 12; // Arrival rate (lamda) = 12 cars/hour
   float pr;         // Probability
   int x;

   printf("Poisson Distribution for ? = %.2f cars/hour:\n", lambda);
   printf("------------------------------------------\n");

   for (x = 0; x <= 15; x++) {
      pr = (exp(-lambda) * pow(lambda, x)) / factorial(x);
      printf("P(X = %2d) = %.6f\n", x, pr);
   }

   return 0;
}
```

OUTPUT

```
C:\Users\Aayush\Desktop\MB

Poisson Distribution for ? = 12.00 cars/hour:
-------------------------------------------
P(X =  0) = 0.000006
P(X =  1) = 0.000074
P(X =  2) = 0.000442
P(X =  3) = 0.001770
P(X =  4) = 0.005309
P(X =  5) = 0.012741
P(X =  6) = 0.025481
P(X =  7) = 0.043682
P(X =  8) = 0.065523
P(X =  9) = 0.087364
P(X = 10) = 0.104837
P(X = 11) = 0.114368
P(X = 12) = 0.114368
P(X = 13) = 0.105570
P(X = 14) = 0.090489
P(X = 15) = 0.072391

--------------------------------
Process exited after 0.09597 seconds with return value 0
Press any key to continue . . .
```

Lab 5: The probabilities of weather conditions (modeled as either rainy or sunny), given the weather on the preceding day, can be represented by a transition matrix:

$$\begin{matrix} \mathbf{0.9} & \mathbf{0.1} \\ \mathbf{0.5} & \mathbf{0.5} \end{matrix}$$

The weather on day 0 is known to be sunny. This is represented by a vector in which the "sunny" entry is 100%, and the "rainy" entry is 0%:

(1 0) .

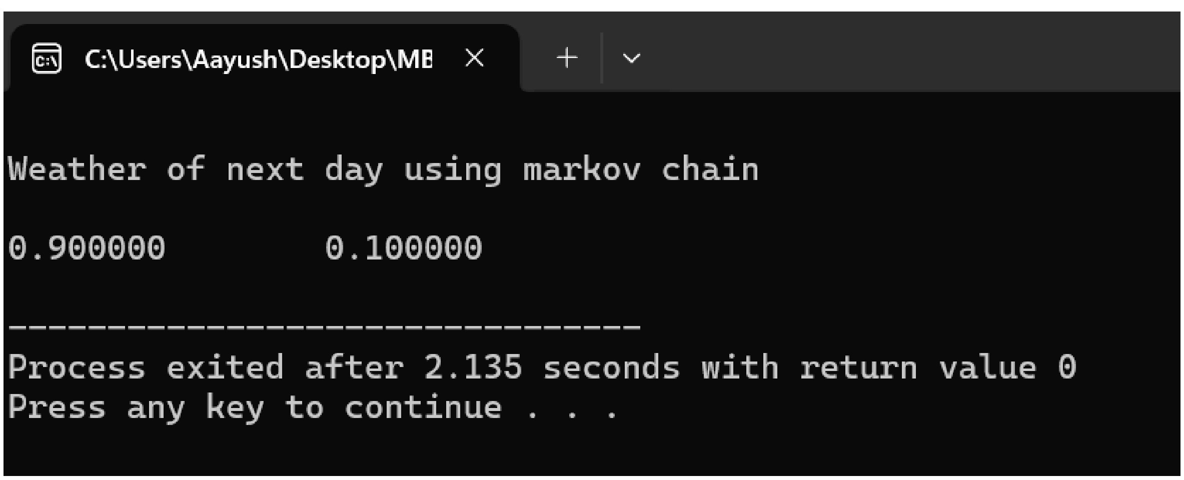
Write a C program to find the weather of the next day by using Markov Chain Method.

Lab-5 solution:

```
#include<stdio.h>

#include<conio.h>


int main() {

    float transMat[2][2]={0.9,0.1,0.5,0.5},vect[1][2]={1,0},result[1][2];

    int i,j,k;

    for(i=0;i<2;i++) {

        for(j=0;j<2;j++) {

            result[i][j]=0;

            for(k=0;k<2;k++) {

                result[i][j]+=vect[i][k]*transMat[k][j];
```

```
                }
            }
        }
        printf("\nWeather of next day using markov chain\n\n");
        for(i=0;i<1;i++) {
            for(j=0;j<2;j++) {
                printf("%f\t",result[i][j]);
            }
            printf("\n");
        }
        getch();
        return 0;
}
```

OUTPUT



Lab 6: What is a random number and what are its properties. WAP in C to generate 100 random numbers using Linear Congruential Method where X0=11, m=100, a = 5 and c = 13.

Answer:

The numbers generated by a process whose outcomes are unpredictable, and which cannot be subsequently reliably reproduced are called random numbers.

Properties of Random Numbers:

1. Uniformity: They are equally probable everywhere
2. Independence: The current value of random variables has no relation with the previous values.

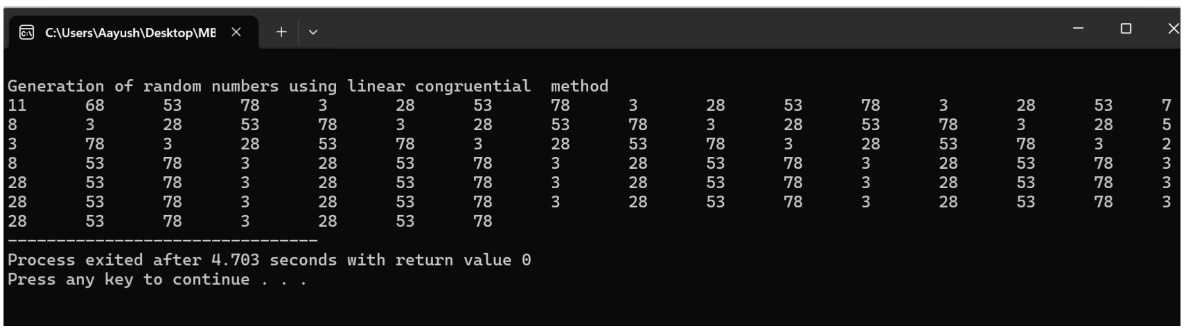
Each random number Ri is an independent sample drawn from a continuous uniform distribution between zero and one.

Solution:

```
#include<stdio.h>

#include<conio.h>

#define SIZE 100

int main() {

	int x[SIZE],i;

	int m=100,a=5,c=13;

	x[0]=11;

	for(i=0;i<100;i++) {

		x[i+1]=(a*x[i]+c)%m;

	}
	printf("\nGeneration of random numbers using linear congruential
method");
```

```
	for(i=0;i<100;i++) {
		printf("%d\t",x[i]);
	}
	getch();
	return 0;
}
```

OUTPUT



Lab 7: WAP in C to generate 100 random numbers using Multiplicative Congruential Method where X0=13, m =1000, a = 15 and c = 7.

Solution

```
#include <stdio.h>
int main() {

    int X0 = 13; // Seed
    int a = 15;  // Multiplier
    int c = 7;   // Increment
    int m = 1000; // Modulus
    int n = 100;  // Number of random numbers to generate
    int random_numbers[n],i;

    random_numbers[0] = X0;
    for (i = 1; i < n; i++) {
        random_numbers[i] = (a * random_numbers[i - 1] + c) % m;
    }

    printf("Generated Random Numbers:\n");
    for ( i = 0; i < n; i++) {
        printf("%d ", random_numbers[i]);
        if ((i + 1) % 10 == 0) {
            printf("\n");
        }
```

```
    }

    return 0;
}
```

OUTPUT