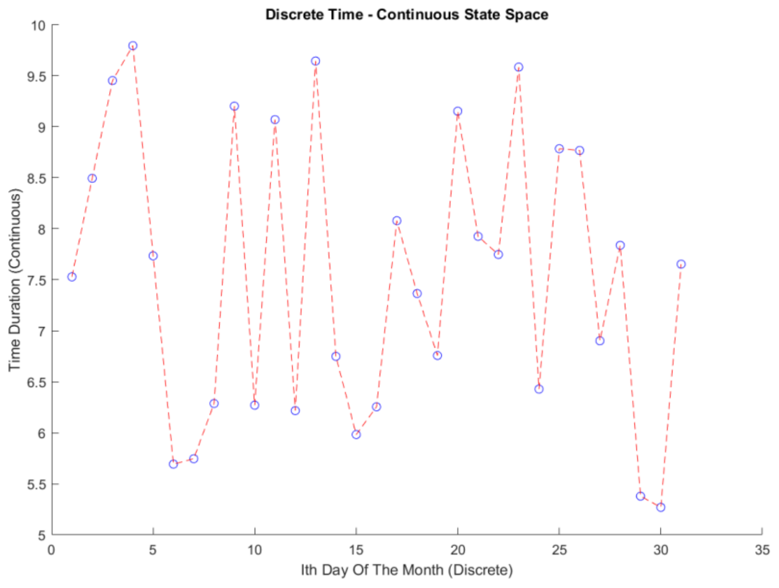
Experiment - 1

Demonstrating a Stochastic Process with Discrete Index Set
a) Continuous State Space
b) Discrete State Space

CODE (A)

```
% Time taken to run a batch job at the nth day of the month
disp("2K22/MC/87")
disp("Experiment - 1(A)")
x = [1:1:31]; % Discrete Time
y = 5 + 5*rand(length(x), 1); % Continuous State Space
figure
scatter(x, y, 'bo');
hold on
plot(x, y, 'r--');
xlabel("Ith Day Of The Month (Discrete)");
ylabel("Time Duration (Continuous)");
title("Discrete Time - Continuous State Space");
```

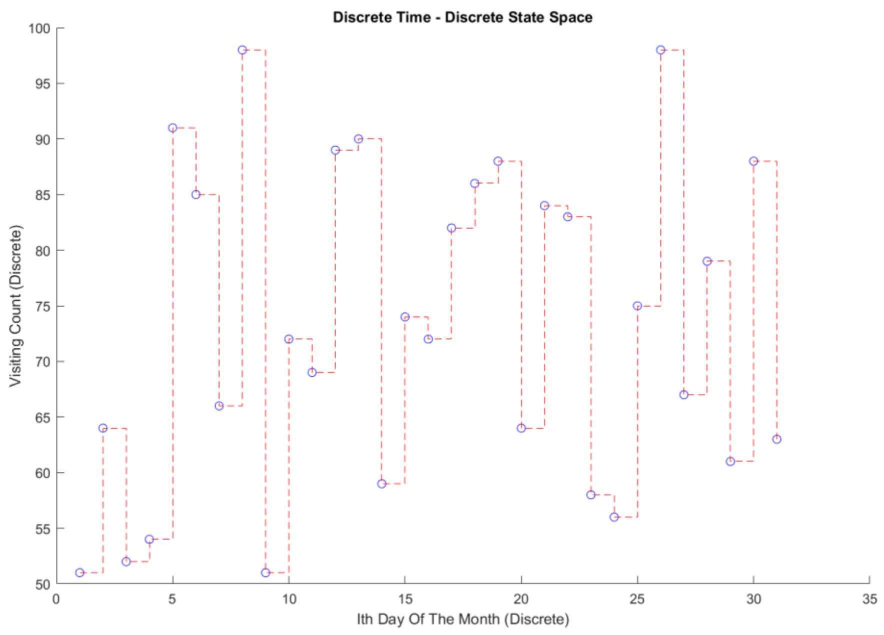
OUTPUT



CODE (B)

```
disp("2K22/MC/87")
disp("Experiment - 1(B)")

x = [1:1:31]; % Discrete Time
y = 50 + randi([0,50], 31, 1); % Discrete State Space
figure
scatter(x, y, 'bo');
hold on
stairs(x, y, 'r--');
xlabel("Ith Day Of The Month (Discrete)");
ylabel("Visiting Count (Discrete)");
title("Discrete Time - Discrete State Space");
```

OUTPUT



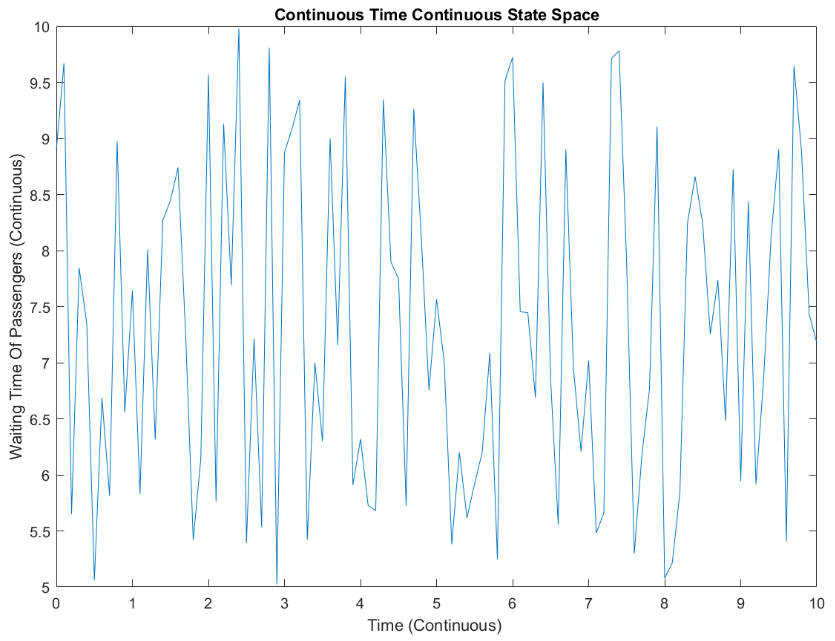
Experiment – 2

Demonstrating a Stochastic Process with Continuous Index Set
a) Continuous State Space
b) Discrete State Space

CODE (A)

```
disp("2K22/MC/87")
disp("Experiment - 2(A)")
x = [0:0.1:10];
y = 5 + 5*rand(length(x), 1);
figure
plot(x,y);
title("Continuous Time Continuous State Space");
xlabel("Time (Continuous) ");
ylabel("Waiting Time Of Passengers (Continuous)");
```

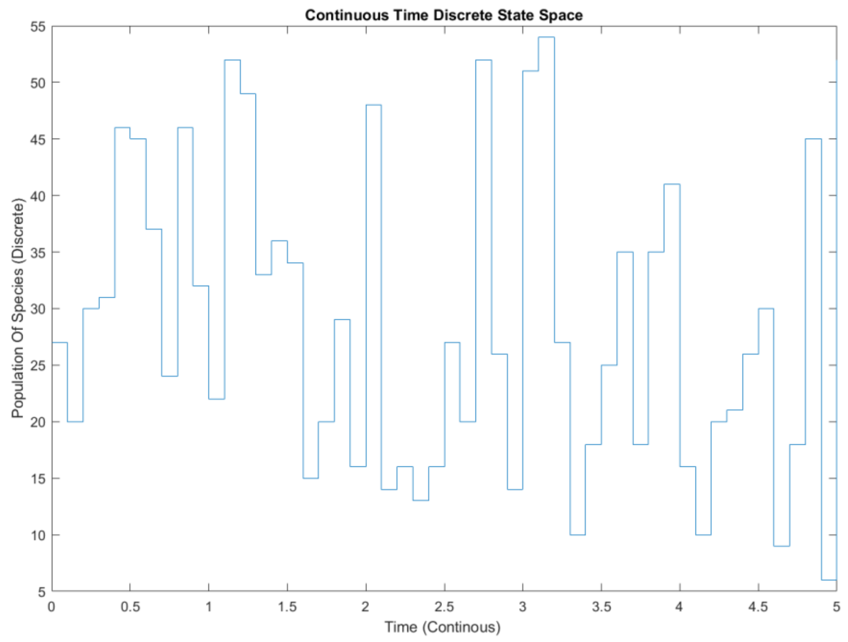
OUTPUT



CODE (B)

```
disp("2K22/MC/87")
disp("Experiment - 2(B)")
x = [0:0.1:5];
y = randi([5 55], length(x), 1);
figure
stairs(x,y);
ylim([5 55]);
title("Continuous Time Discrete State Space");
xlabel("Time (Continous) ");
ylabel("Population Of Species (Discrete)");
```

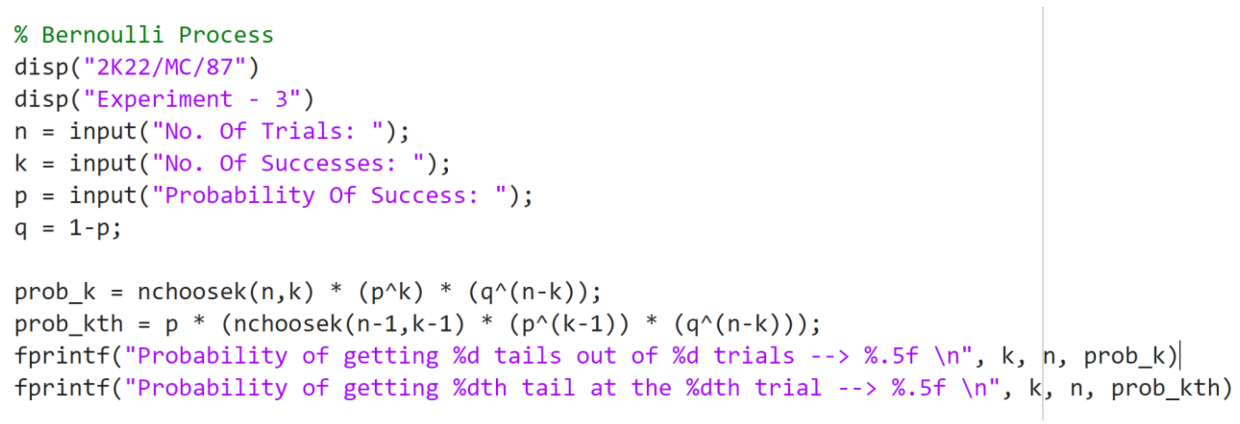
OUTPUT



Experiment - 3

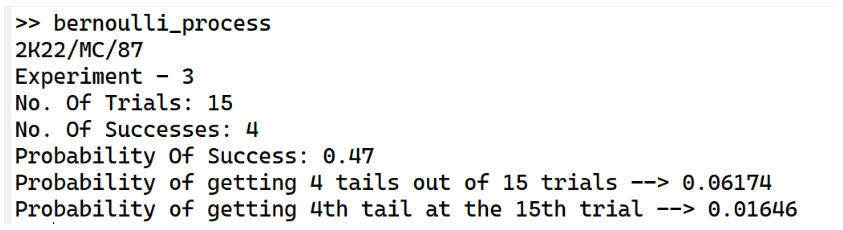
Demonstrating Bernoulli Process. Write a program to find the probability the in case a Bernoulli Process
a) out of n trials k are successes
b) kth success occurs at the nth trial

CODE



```
% Bernoulli Process
disp("2K22/MC/87")
disp("Experiment - 3")
n = input("No. Of Trials: ");
k = input("No. Of Successes: ");
p = input("Probability Of Success: ");
q = 1-p;

prob_k = nchoosek(n,k) * (p^k) * (q^(n-k));
prob_kth = p * (nchoosek(n-1,k-1) * (p^(k-1)) * (q^(n-k)));
fprintf("Probability of getting %d tails out of %d trials --> %.5f \n", k, n, prob_k)
fprintf("Probability of getting %dth tail at the %dth trial --> %.5f \n", k, n, prob_kth)
```

OUTPUT

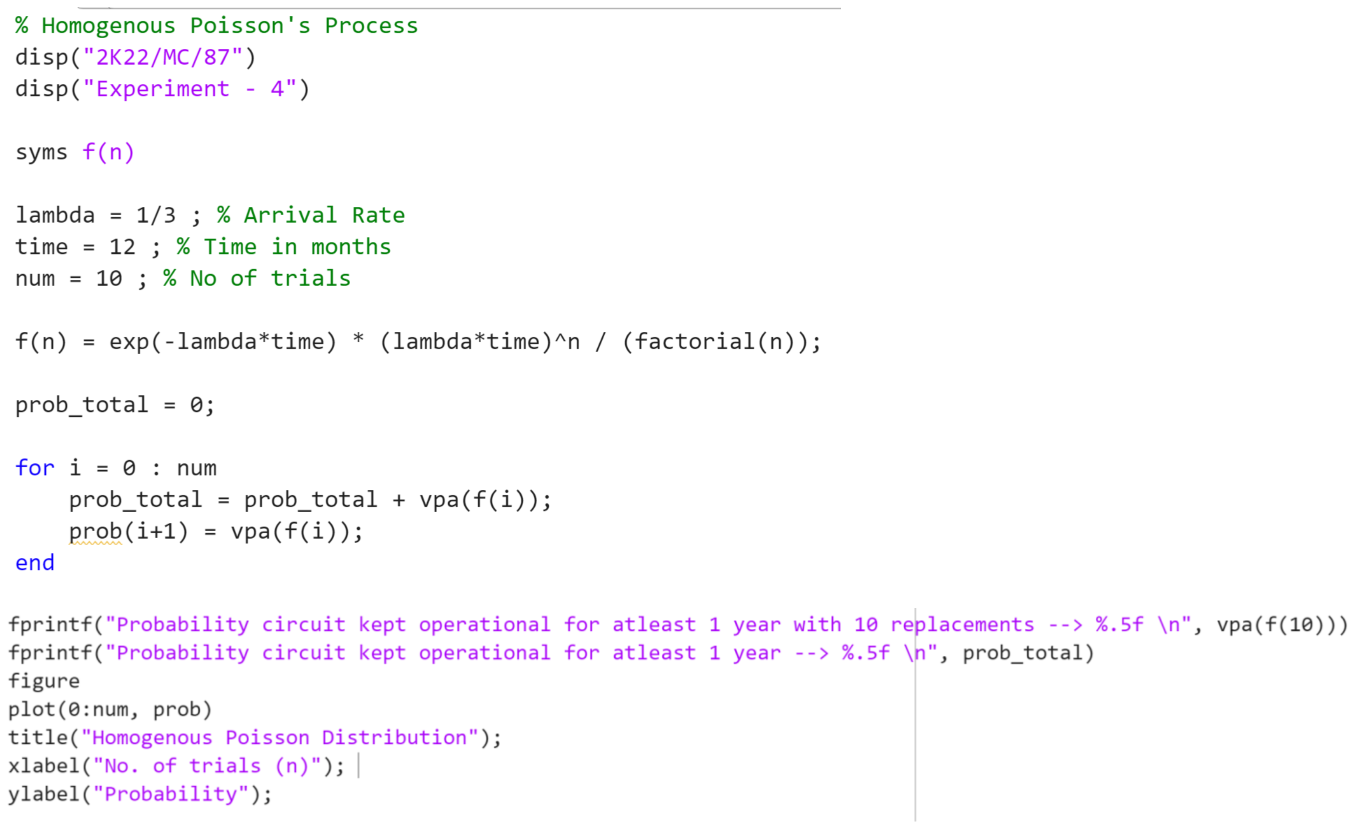


```
>> bernoulli_process
2K22/MC/87
Experiment - 3
No. Of Trials: 15
No. Of Successes: 4
Probability Of Success: 0.47
Probability of getting 4 tails out of 15 trials --> 0.06174
Probability of getting 4th tail at the 15th trial --> 0.01646
```

Experiment - 4

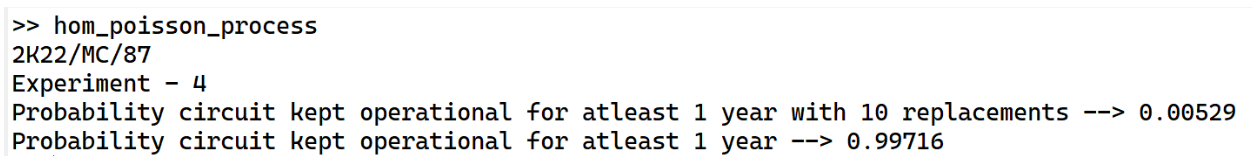
Demonstrating Poisson Process. Write a program to find the probability that in case of Poisson process with arrival rate λ, in a length of time t there are exactly k arrivals

CODE

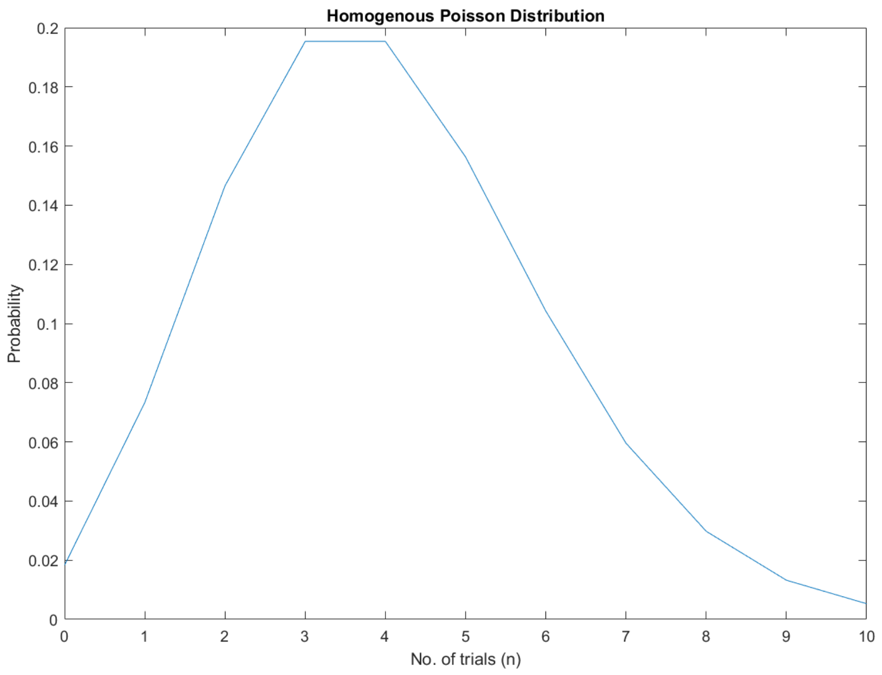


```
% Homogenous Poisson's Process
disp("2K22/MC/87")
disp("Experiment - 4")

syms f(n)

lambda = 1/3 ; % Arrival Rate
time = 12 ; % Time in months
num = 10 ; % No of trials

f(n) = exp(-lambda*time) * (lambda*time)^n / (factorial(n));

prob_total = 0;

for i = 0 : num
    prob_total = prob_total + vpa(f(i));
    prob(i+1) = vpa(f(i));
end

fprintf("Probability circuit kept operational for atleast 1 year with 10 replacements --> %.5f \n", vpa(f(10)))
fprintf("Probability circuit kept operational for atleast 1 year --> %.5f \n", prob_total)
figure
plot(0:num, prob)
title("Homogenous Poisson Distribution");
xlabel("No. of trials (n)");
ylabel("Probability");
```

OUTPUT



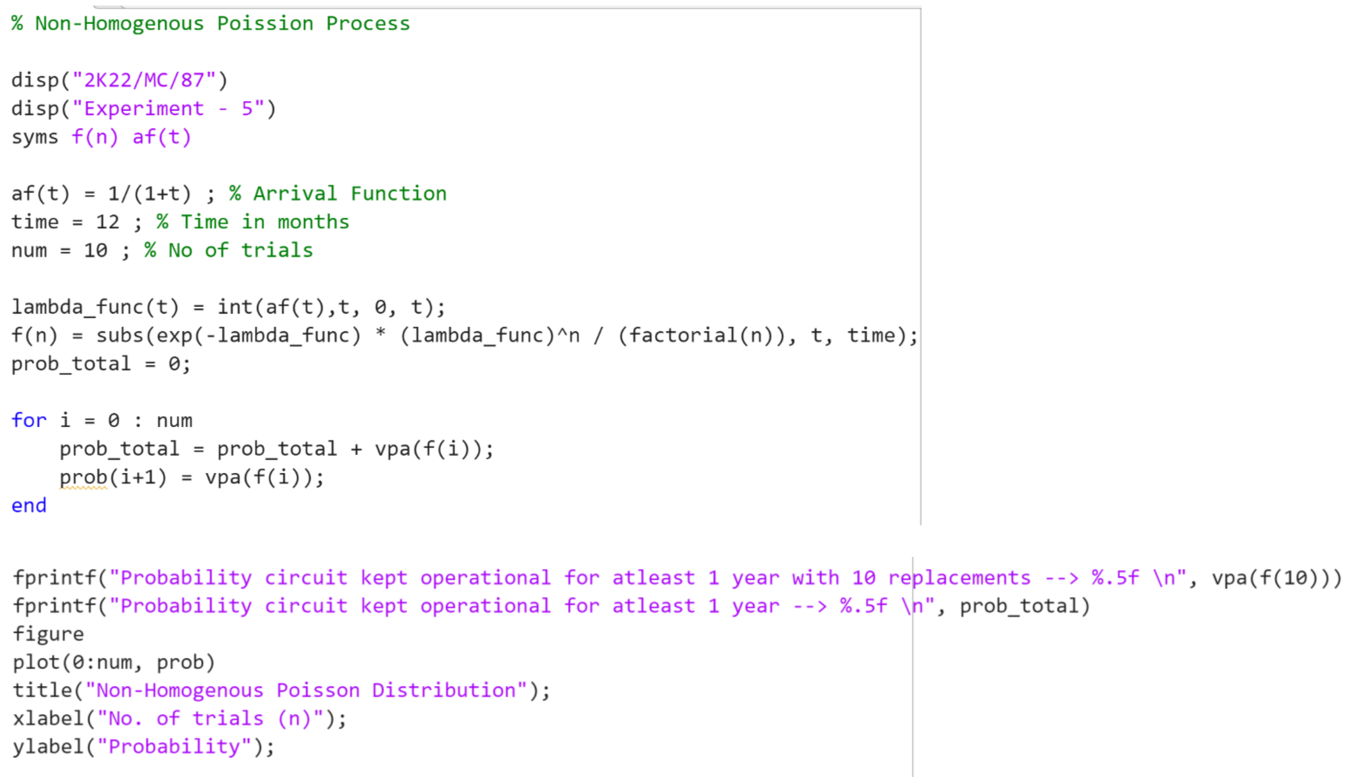
```
>> hom_poisson_process
2K22/MC/87
Experiment - 4
Probability circuit kept operational for atleast 1 year with 10 replacements --> 0.00529
Probability circuit kept operational for atleast 1 year --> 0.99716
```


Homogenous Poisson Distribution
Probability
No. of trials (n)
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0
1
2
3
4
5
6
7
8
9
10

Experiments - 5

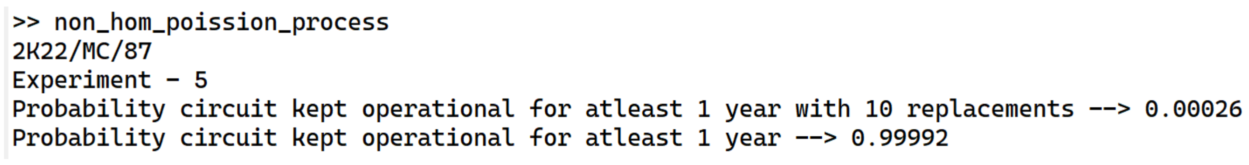
Demonstrating Poisson Process (Non-Homogeneous). Write a program to find the probability that in case of Poisson Process with arrival rate λ(t) in a length of time t there are exactly k arrivals

CODE

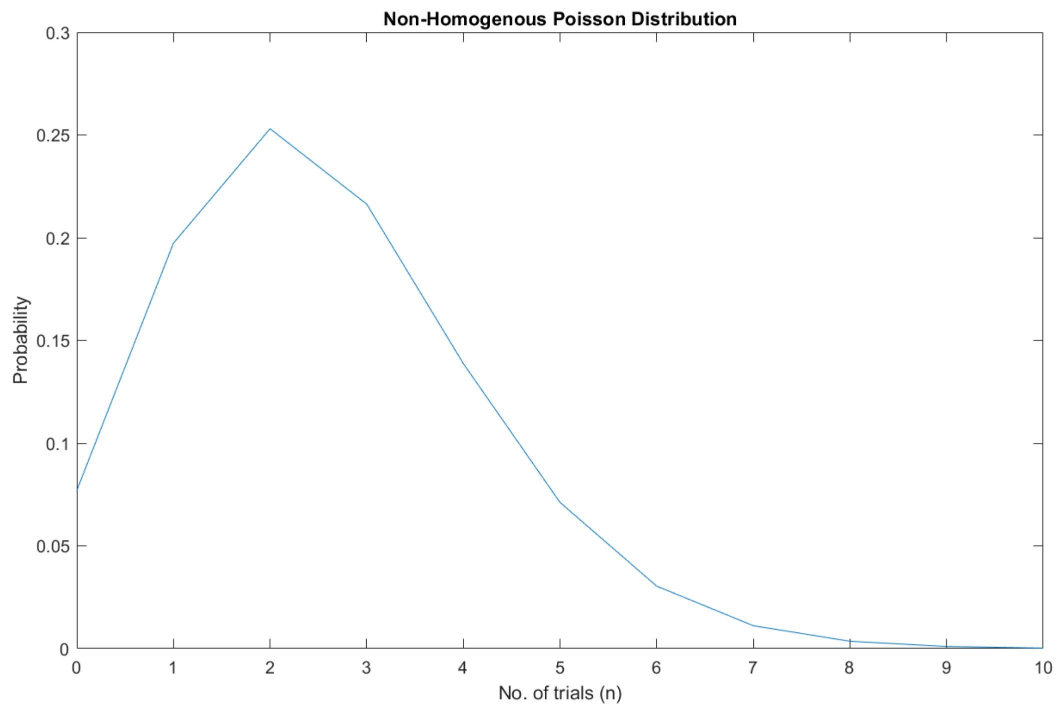


```
% Non-Homogenous Poission Process

disp("2K22/MC/87")
disp("Experiment - 5")
syms f(n) af(t)

af(t) = 1/(1+t) ; % Arrival Function
time = 12 ; % Time in months
num = 10 ; % No of trials

lambda_func(t) = int(af(t),t, 0, t);
f(n) = subs(exp(-lambda_func) * (lambda_func)^n / (factorial(n)), t, time);
prob_total = 0;

for i = 0 : num
    prob_total = prob_total + vpa(f(i));
    prob(i+1) = vpa(f(i));
end

fprintf("Probability circuit kept operational for atleast 1 year with 10 replacements --> %.5f \n", vpa(f(10)))
fprintf("Probability circuit kept operational for atleast 1 year --> %.5f \n", prob_total)
figure
plot(0:num, prob)
title("Non-Homogenous Poisson Distribution");
xlabel("No. of trials (n)");
ylabel("Probability");
```

OUTPUT



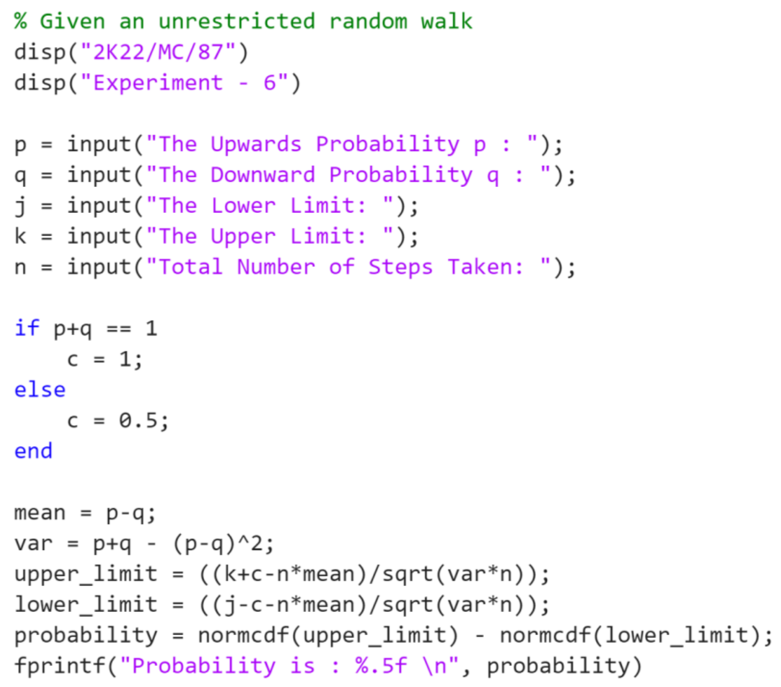
```
>> non_hom_poission_process
2K22/MC/87
Experiment - 5
Probability circuit kept operational for atleast 1 year with 10 replacements --> 0.00026
Probability circuit kept operational for atleast 1 year --> 0.99992
```


Non-Homogenous Poisson Distribution
Probability
No. of trials (n)
0.3
0.25
0.2
0.15
0.1
0.05
0
0
1
2
3
4
5
6
7
8
9
10

Experiment - 6

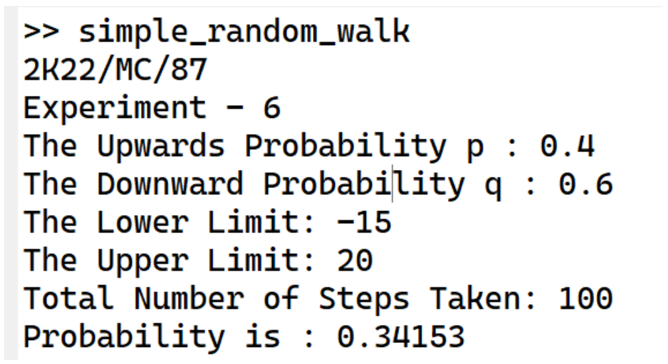
Demonstrating Simple Random Walk. Write a program to find the probability that in case of an unrestricted random walk, at the nth instant particle lies between two specified limits

CODE



```
% Given an unrestricted random walk
disp("2K22/MC/87")
disp("Experiment - 6")

p = input("The Upwards Probability p : ");
q = input("The Downward Probability q : ");
j = input("The Lower Limit: ");
k = input("The Upper Limit: ");
n = input("Total Number of Steps Taken: ");

if p+q == 1
    c = 1;
else
    c = 0.5;
end

mean = p-q;
var = p+q - (p-q)^2;
upper_limit = ((k+c-n*mean)/sqrt(var*n));
lower_limit = ((j-c-n*mean)/sqrt(var*n));
probability = normcdf(upper_limit) - normcdf(lower_limit);
fprintf("Probability is : %.5f \n", probability)
```

OUTPUT

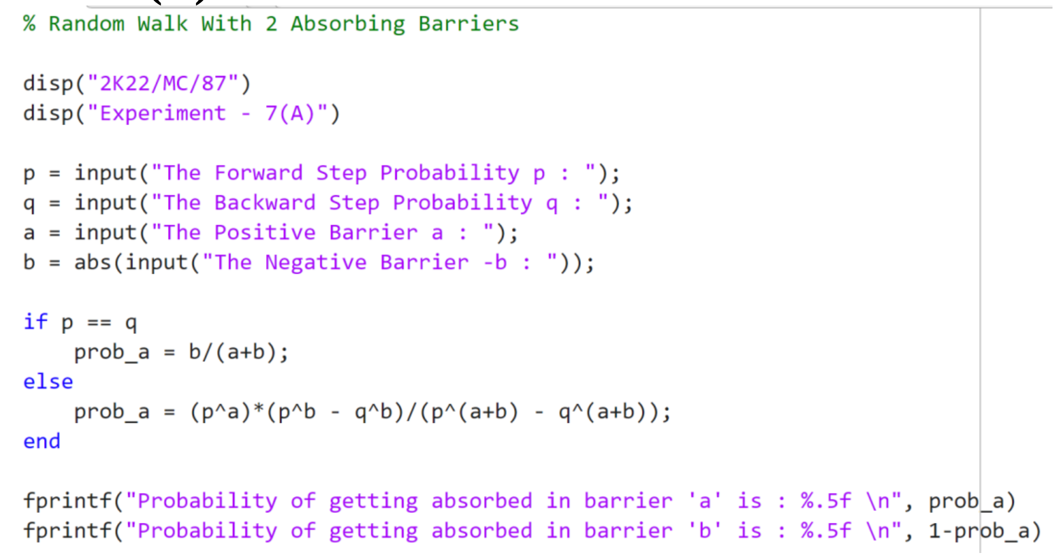


```
>> simple_random_walk
2K22/MC/87
Experiment - 6
The Upwards Probability p : 0.4
The Downward Probability q : 0.6
The Lower Limit: -15
The Upper Limit: 20
Total Number of Steps Taken: 100
Probability is : 0.34153
```

Experiment - 7

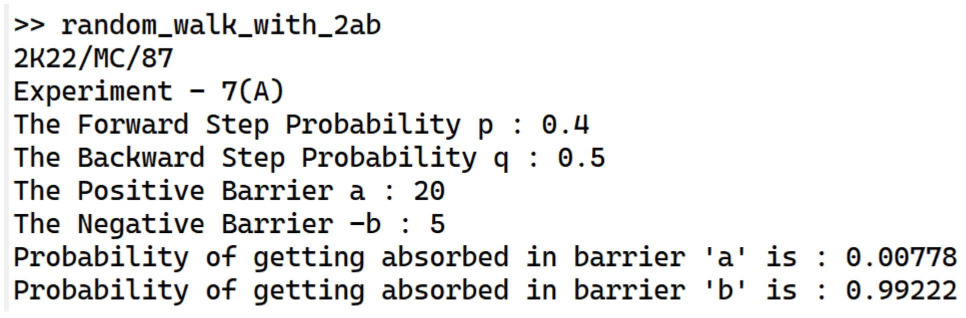
Demonstrating Simple Random Walk. Write a program to find the probability that in case of
a) A random walk with 2 absorbing barriers, the probability of absorption at a specific barrier
b) A random walk with 2 reflecting barriers, steady state probability distribution for the possible states

CODE (A)



```
% Random Walk With 2 Absorbing Barriers

disp("2K22/MC/87")
disp("Experiment - 7(A)")

p = input("The Forward Step Probability p : ");
q = input("The Backward Step Probability q : ");
a = input("The Positive Barrier a : ");
b = abs(input("The Negative Barrier -b : "));

if p == q
    prob_a = b/(a+b);
else
    prob_a = (p^a)*(p^b - q^b)/(p^(a+b) - q^(a+b));
end

fprintf("Probability of getting absorbed in barrier 'a' is : %.5f \n", prob_a)
fprintf("Probability of getting absorbed in barrier 'b' is : %.5f \n", 1-prob_a)
```

OUTPUT



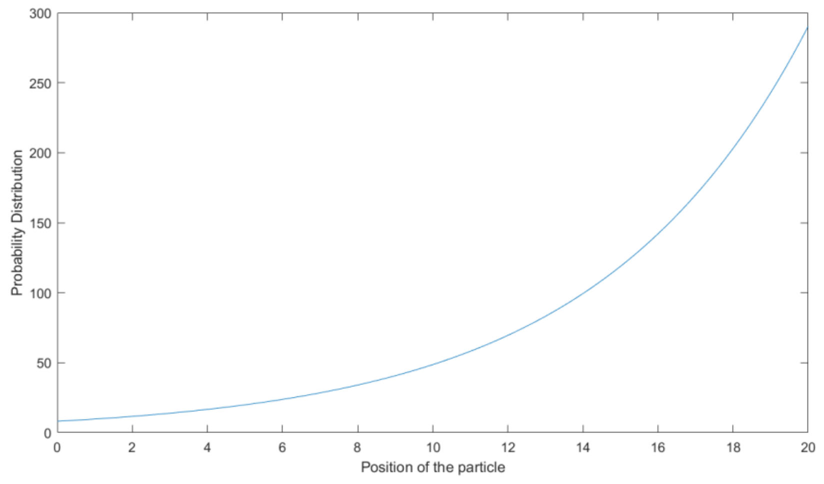
```
>> random_walk_with_2ab
2K22/MC/87
Experiment - 7(A)
The Forward Step Probability p : 0.4
The Backward Step Probability q : 0.5
The Positive Barrier a : 20
The Negative Barrier -b : 5
Probability of getting absorbed in barrier 'a' is : 0.00778
Probability of getting absorbed in barrier 'b' is : 0.99222
```

CODE (B)

```
% Given an random walk with 2 reflecting barriers

disp("2K22/MC/87")
disp("Experiment - 7(B)")
p = input("The Forward Step Probability p : ");
q = input("The Backward Step Probability q : ");
% Assuming on absorbing barrier to be the origin
a = input("The Position of the reflecting barrier a : ");

if p==q
    pi(1) =1/a+1;
else
    pi(1) = (1-(p/q))*(1-(p/q)^(a+1));
end

for i=1:100
    k = (a*i)/100;
    pi(i+1) = pi(1) * ((p/q)^k); % k = 0,1,2,......,a-1,a
end
plot(0:a/100:a, pi)
xlabel("Position of the particle");
ylabel("Probability Distribution");
```

OUTPUT

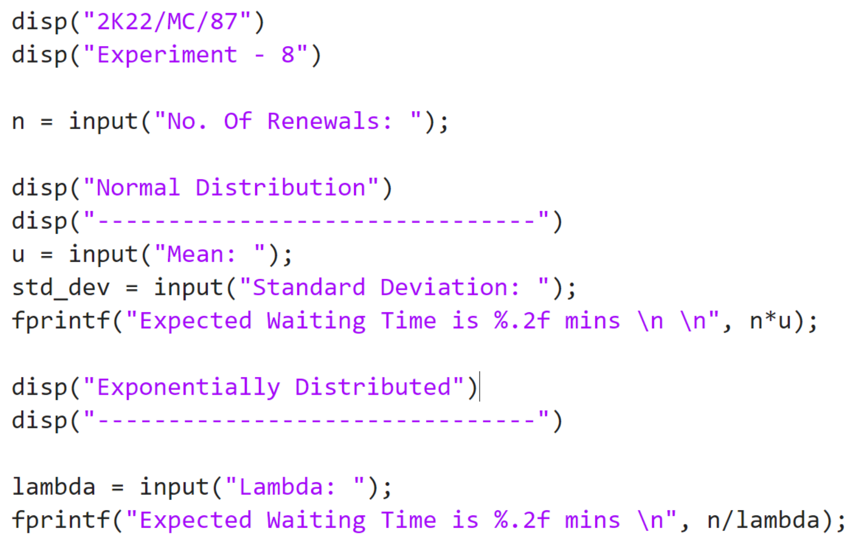
```
>> random_walk_with_2rf
2K22/MC/87
Experiment - 7(B)
The Forward Step Probability p : 0.55
The Backward Step Probability q : 0.46
The Position of the reflecting barrier a : 20
```



Experiment - 8

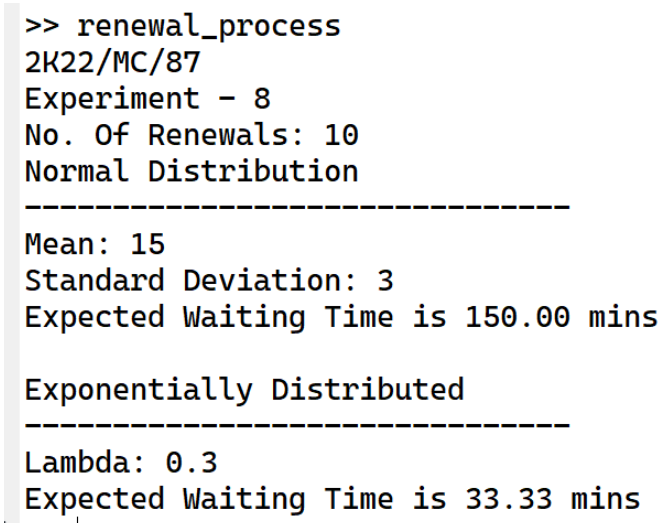
Demonstrating Renewal Process. Write a program to find the expected waiting time until the nth renewal in case of a renewal process with renewal cycle length distributed
a) Normally with mean μ standard deviation σ (μ >3 σ)
b) Exponentially with parameter λ

CODE



```
disp("2K22/MC/87")
disp("Experiment - 8")

n = input("No. Of Renewals: ");

disp("Normal Distribution")
disp("-------------------------------")
u = input("Mean: ");
std_dev = input("Standard Deviation: ");
fprintf("Expected Waiting Time is %.2f mins \n \n", n*u);

disp("Exponentially Distributed")
disp("-------------------------------")

lambda = input("Lambda: ");
fprintf("Expected Waiting Time is %.2f mins \n", n/lambda);
```

OUTPUT

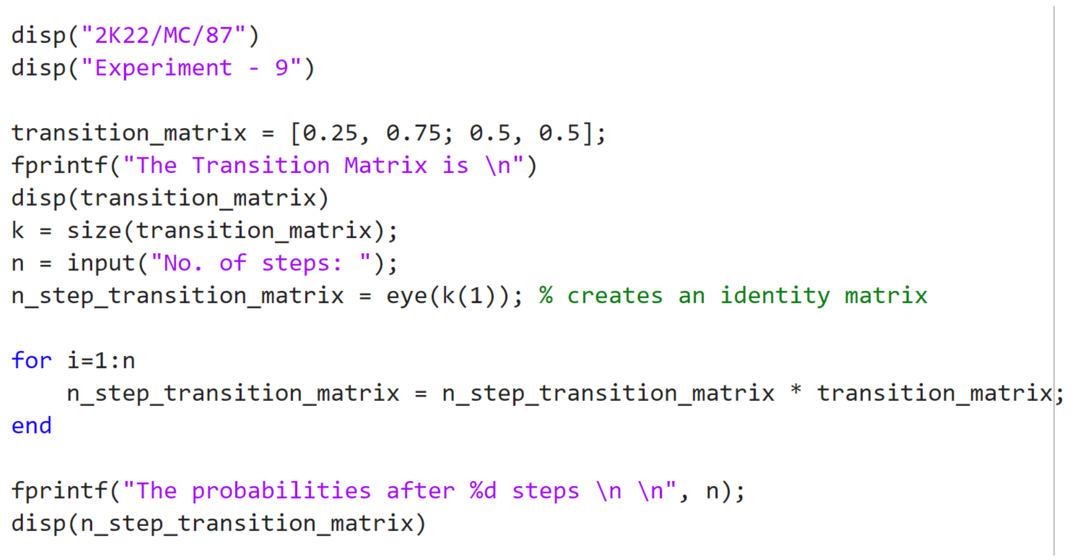


```
>> renewal_process
2K22/MC/87
Experiment - 8
No. Of Renewals: 10
Normal Distribution
-------------------------------
Mean: 15
Standard Deviation: 3
Expected Waiting Time is 150.00 mins

Exponentially Distributed
-------------------------------
Lambda: 0.3
Expected Waiting Time is 33.33 mins
```

Experiment - 9

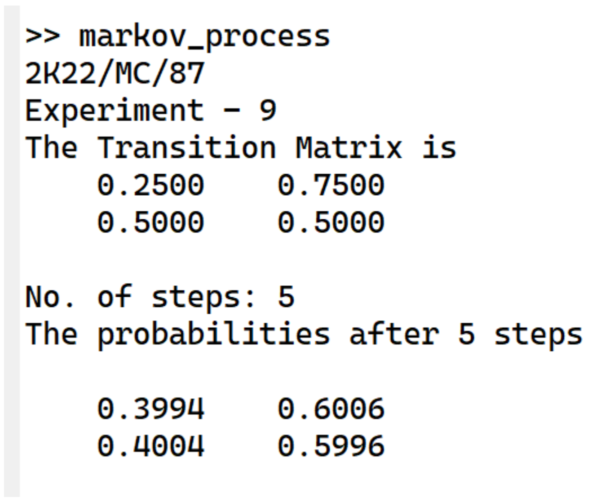
Demonstrating Markov Chain. Write a program to find the n-step transition probability matrix in case of Markov Chain

CODE



```
disp("2K22/MC/87")
disp("Experiment - 9")

transition_matrix = [0.25, 0.75; 0.5, 0.5];
fprintf("The Transition Matrix is \n")
disp(transition_matrix)
k = size(transition_matrix);
n = input("No. of steps: ");
n_step_transition_matrix = eye(k(1)); % creates an identity matrix

for i=1:n
    n_step_transition_matrix = n_step_transition_matrix * transition_matrix;
end

fprintf("The probabilities after %d steps \n \n", n);
disp(n_step_transition_matrix)
```

OUTPUT



```
>> markov_process
2K22/MC/87
Experiment - 9
The Transition Matrix is
    0.2500    0.7500
    0.5000    0.5000

No. of steps: 5
The probabilities after 5 steps

    0.3994    0.6006
    0.4004    0.5996
```