Exp1-

```
clc;
close all;
n = input('no. of bits=');
l = (2^n) - 1;
ff=[];
ff(1)=input('no.1 : ');
ff(2)=input('no.2 : ');
ff(3)=input('no.3 : ');
ff(4)=input('no.4 : ');
for i=1:15
temp = ff(1);
ff(1)=xor(ff(4),ff(1));
ff(4)=ff(3);
ff(3)=ff(2);
ff(2)=temp;
disp(ff(4));
end
```

Exp2-

```
clc;
clear all;
close all;
total_bandwidth = 40e6;
channel_bandwidth = 30e3;
disp(channel_bandwidth)
cell_radius = 2;
area_circle = 270;
cluster_size = input('num_cluster');

num_channels =round(total_bandwidth / (2*channel_bandwidth));
area_hex = round(3 * sqrt(3) / 2 * cell_radius^2 );
num_cells = round( area_circle / area_hex);

disp(['Number of channels per cell: ' num2str(num_channels)]);
disp(['Number of cells in the system: ' num2str(num_cells)]);
disp(' ');

cluster_sizes = [cluster_size];
for i = 1:length(cluster_sizes)
    K = cluster_sizes(i);
    cluster_repetition = num_cells / K;
    system_capacity(i) = num_channels * cluster_repetition;
    disp(['System capacity for cluster size ' num2str(K) ': ' num2str(system_capacity(i)) ' channels']);
end
```

Exp3-

```
clc;
close all;
clear all;
n= input ('enter the value of n :');
SIR = 15;
powl=2/n;
SIR=power(10,SIR/10)
N=round((power(6*SIR,powl))/3);
fprintf ('N=% d' ,N);
```

```
disp('');
valid=0;
for i = 0:1:5
    for j = 0:1:5
        x=power(i,2)+power(j,2)+i*j;
        if x ==N
            valid=1;
            break;
        end
    end
end
arr=zeros();
p = 1
if valid==0
    for i = 0:1:5
        for j = 0:1:5
            x=power(i,2)+power(j,2)+i*j;
            arr(p)=x;
            p = p + 1;
        end
    end
end
sorted_arr=sort(arr);
for i = 1:1:15
    if sorted_arr(i) > N
        N=sorted_arr(i);
        break;
    end
end
fprintf('verification N=% d' ,N)
```

Exp4-

```
clc;
close all;
clear all;
N=4;
r=1.56;
lambda=1;
No_of_channels=80;
Au=0.029;
Pr=0.05;
A=13;
C=No_of_channels/N;
cell_area=(3*sqrt(3)*r*r)/2;
Nc=A/Au;
H=(Au/lambda)*60*60;
Ns=round(Nc/cell_area);
fprintf('Number of channela per cell:%d',Ns)
j=1;
for i=1:1:3
    t=input('\nEnter the dealy time: ');
    Pr1=exp((-(C-A)*t)/H);
    j=j+1;
    fprintf('%d Probability that a delayed call will have to wait for more than %d sec:',j,t)
    disp(Pr1)
    j=j+1;
    Pr_t=(Pr*Pr1)*100;
    fprintf('%d Probability that a call will be delayed for more than %d sec:',j,t)
```

```
   disp(Pr_t)
end
```

Exp5-

```
clc;
clear all;
close all;
fc=input('Enter first carrier frequency :');
fc2=input('Enter Second carrier frequency :');
ht= 100;
hr= 4;
r=2:2:26;
ar= 3.2*log((11.75*hr)^2)-4.97;
Lph=68.75+26.16*log(fc)-13.82*log(ht)-ar+(44.9-6.55*log(ht))*log(r);
ar2= 3.2*log((11.75*hr)^2)-4.97;
Lph2=68.75+26.16*log(fc2)-13.82*log(ht)-ar2+(44.9-6.55*log(ht))*log(r);
plot(r,Lph,r,Lph2);
```

Exp6-

```
clc;
close all;
clear all;
fc = 900000000;
v = 70*1000/3600;
disp('Velocity = ')
disp(v)
lambda = 300000000/fc;
disp('lambda = ')
disp(lambda)
angle = input('Enter the input angle');
if angle == 0
 disp('Mobile user moving towards BS = ')
elseif angle == pi
 disp('Mobile user moving away from the BS = ')
elseif angle == pi/6
 disp('Mobile user moving towards BS at 30 degree = ')
elseif angle == pi/3
 disp('Mobile user moving towards BS at 60 degree = ')
elseif angle == pi - pi/3
 disp('Mobile user moving away from the BS at 60 degree = ')
elseif angle == pi - pi/6
 disp('Mobile user moving away from the BS at 30 degree = ')
else
 disp('Invalid angle')
end
fd = (v*cos(angle))/lambda;
frr = fc + fd;
fr = rat(frr);
disp('Received carrier frequency = ')
disp(fr)
```

Exp7-

```
clc
clear all;
close all;
u1 = input('Enter Polar Form of User Data 1: ');
u2 = input('Enter Polar Form of User Data 2 : ');
pn1 = input('Enter Polar Form of PN1: ');
```

```
pn2 = input('Enter Polar Form of PN2: ');
disp('multiplication of user data and pn sequence')
pn11 = u1(1)* pn1;
pn12 = u1(2)* pn1;
pn13= u1(3)* pn1;
pn14 = u1(4)* pn1;
disp([pn11 pn12 pn13])
pn21 = u2(1)* pn2;
pn22 = u2(2)* pn2;
pn23= u2(3)* pn2;
pn24 = u2(4)* pn2;
disp([pn21 pn22 pn23])
disp('then add the new user1 and user2 bits')
a =pn11+pn21 ;
b =pn12+pn22;
c =pn13+pn23 ;
d =pn14+pn24;
disp('for first user')
R1_1= a.* pn1;
R1_2= b.* pn1;
R1_3= c.* pn1;
R1_4 =d.* pn1;
disp([R1_1 R1_2 R1_3 R1_4])
disp('for second user')
R2_1= a.* pn2;
R2_2= b.* pn2;
R2_3= c.* pn2;
R2_4 =d.* pn2;
disp([R2_1 R2_2 R2_3 R2_4])
disp('final answer for user1')
F1_1 = sum(R1_1)/4;
F1_2 = sum(R1_2)/4;
F1_3 = sum(R1_3)/4;
F1_4 = sum(R1_4)/4;
disp([F1_1 F1_2 F1_3 F1_4]);
disp('final answer for user2')
F2_1 = sum(R2_1)/4;
F2_2 = sum(R2_2)/4;
F2_3 = sum(R2_3)/4;
F2_4 = sum(R2_4)/4;
disp([F2_1 F2_2 F2_3 F2_4]);
```

Exp8-

```
clc;
close all;
clear all;
H1 = input('Enter input (1/0):');
disp('H1 matrix is = ')
disp(H1)
H2 = [H1 H1;H1 not(H1)];
disp('H2 matrix is = ')
disp(H2)
H4 = [H2 H2;H2 not(H2)];
disp('H4 matrix is = ')
disp(H4)
H8 = [H4 H4;H4 not(H4)];
disp('H8 matrix is = ')
disp(H8)
disp('Checking for Properties:')
disp('Orthogonality:')
c_b = 0;
dc_b = 0;
t = input('Enter n for nxn matrix');
for i = 1:t
 j = 1;
 if t == 4
 if H4(i,j) == H4(i,j+1);
 c_b = c_b + 1;
 else
 dc_b = dc_b + 1;
 end
 elseif t == 8
 if H8(i,j) == H8(i,j+1);
 c_b = c_b + 1;
 else
 dc_b = dc_b + 1;
 end
 else
 disp('Value of n is invalid:')
 end
end
disp('Number of continuous bits = ')
disp(c_b)
disp('Number of discontinuous bits = ')
disp(dc_b)
h = (c_b - dc_b)/t;
disp(h)
if h == 0
 disp('Hence orthogonality is proved')
else
 disp('Hence orthogonality is not satified')
end
```

Exp10-

```
clc;
clear all;
close all;
L=3;
N=256;
SNR = 0:25;
y = db2pow(SNR);
a = nchoosek(5,3);
b = 2*N*y;
BER1= a*((1./b).^2);
L=4;
a = nchoosek(7,4);
b = 2*N*y;
BER2= a*((1./b).^2);
y=0:2:25;
p = plot(SNR,BER1,SNR,BER2,'--');
xlabel('SNR (dB)');
ylabel('Bit Error Rate');
```