HAMMING CODE

```
cle;clear all; close all;
n=7%# of codeword bits per block
k= 4%# of message bits per block
A=[1 1 1;1 1 0;1 0 1;0 1 1 ]:%Parity submatrix-Need binary(decimal combination of
7,6,5,3)
G=[eye(k) A ]%oGenerator matrix
H=[A' eye(n-k) ]%Parity-check matnıx
% ENCODER%
msg =[111 1] %Message block vector-change to any 4 bit sequence
code = mod(msg*G,2)%Encode message

code(2)=~code(2);

recd =code        %Received codeword with error

syndrome = mod(recd * H',2)
%Find position of the error in codeword (index)
find= 0;
for ii= 1:n
if~find
        errvect=zeros(1,n);
        errvect(ii) = 1;
        search = mod(errvect * H',2);
        if search == syndrome
        find= l;
        index=ii;
        end
```

```
    end
end
disp(['Position of error in codeword=',num2str(index)]);
correctedcode= recd;
corectedcode(index)= mod(recd(index)+1,2)%Corrected codeword
%Strip offparity bits
msg_decoded=correctedcode;
msg_decoded=msg_decoded(1:4)
```

CONVOLUTION CODE

```
constraint_length=3;
generator_polynomials=[5,7];
code_rate=1/3;
t=poly2trellis(constraint_length,generator_polynomials);
input_data=[1 0 1 0 1];
encoded_data=convenc(input_data,t);
received_data=encoded_data;
tb_depth=length(input_data);
decoded_data=vitdec(received_data,t,tb_depth,'trunc','hard');
disp('original data:');
disp(input_data);
disp('Encoded data:');
disp(encoded_data);
disp('decoded data:');
disp(decoded_data);
```

HOFFMANN CODING

```
clc;clear ali; close all;
symbol =[1:5]; % Distinct data symbols appearing in
p=[0.4 0.2 0.2 0.1 0.1]; % Probability of each data
[dict,avglen]-huffmandict(symbol,p)
samplecode - dict[5,2) % Codeword for fifth signal value ($ ):s p? C r( |t
diet(l.:]dict/2.:]dict(3,:)dict/4.)dien(5.)
hcode = huffmanenco(symbol,dict);
dhsig= huffmandeco(hcode,dict); % Decode the code.
disp('encoded msg:');
disp(hcode);
disp('decoded msg:");
disp(dhsig);
code length=length(hcode)
#-Sus
sum=0;
for m=1:5
        H=sum+(p(m)*log2(1/p(m)));
end
disp(H=");
disp(H);
Eticicney=(H/avglen)*100
```

BUCKET

```
#include<stdio.h>
#include<stdlib.h>
#include <stdio.h>
int main(int x, int .y)
```

```
{
        if(x=y)
                return x;
        else
                return y;
}
int main()
{
int drop=0, count=0, inp[25];
int mini, nsec, cap, i, process;
printf("'n Enter the Bucket Size: ");
scanf("%d",&cap);
printf("In Enter the Operation Rate: ");
scanf("%d",&process);
printf("n Enter the no. of Seconds you want to Stimulate: ");
scanf("%d",&nsec);
for(i-0;i<nsee;i++)
{
        printf("In Enter the Size of the Packet entering at %d sec: ",i+1);
        scanf("'%d",&inp[i]);
}
printf("\nSecond|PacketRecieved|PacketSentPacketLeftlPacket Dropped\n");
printf("-----------------------------------------------\n”);.
for(i=0;i<nsec;i++)
{
count+=inp[i];
if(count>cap)
{
drop=count-cap;
```

```
count=cap;
}
printf("%d",i+1);
printf("\t%d",inp[i]);
mini=min(count,process);
print("\t\t%d",mini);
count=count-mini;
printf("\t\t%d",count);
printf("\t\t%din",drop);
drop=0;
}
for(;count!=0;i++)
{
if(count>cap)
{
        drop=count-cap;
        count=cap;
}
printf("%d",i+1);
printf("\t0");
mini=min(count, process);
printf("\t\t %d", mini);
count=count-mini;
printf('\t\t%d", count);
printf("\t\t%d\n", drop);
}
}
```

SLIDING WINDOW

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
int w,i,f.frames[50];
printf("Enter window size: ");
scanf("%d",&w);
printf("/n Enter number of frames to transmit: ');
scanf("%d", &f);
printf("/nEnter %d frames: ",f);
for(i=1; i<=f; i++)
scanf("%d",&frames[i]);
printf("/n With sliding window protocol the frames will be sent in the following manner (assurning no corruption of frames)\n\n");
printf(" After sending %d frames at each stage sender waits for
acknowledgement sent by the receiverln\n",w);
for(i=1;i<=f;i++)
{
if(i%w==0)
{
printf("%d\n",frames[i]);
printf("Acknowledgement of above frames sent is received by sender\n\n");
}
else
printf("%d ",frames[i]);
}
if(f%w!=0)
```

```
printf('InAcknowledgement of above frames sent is received by
senderin");
return 0;
}
```

njjjj