# THIS CODES ARE ONLY PERFROMED IN C COMPILER

# EXP 7 :-DETERMINATION OF ENTROPIES & INFORMATION & INFORMATION RATE

# PROGRAM:

#include <stdio.h> #include <math.h> #include <string.h> #include <time.h>

int main() {

char string1[100]; int i, s;

float p, l, m, h = 0.0, k, r, R;

int z[256] = {0}; // Assuming ASCII characters

printf("Enter the string: "); fgets(string1, sizeof(string1), stdin);

string1[strcspn(string1, "\n")] = '\0'; // Remove newline from fgets

clock\_t start, end; start = clock(); end = clock();

r = (float)(end - start) / CLOCKS\_PER\_SEC; l = strlen(string1);

printf("\nLength of string = %f\n", l); for (i = 0; i < l; i++) {

s = (int)string1[i];

z[s] = z[s] + 1; // Count frequency of each character

}

printf("\nSymbol\t Frequency\t Probability\t Information\n"); for (i = 0; i < 256; i++) { // Check all ASCII characters

if (z[i] != 0) {

printf("%c\t\t %d\t\t\t %f\t\t %f bits\n", i, z[i], (float)z[i] / l, log2(l / (float)z[i])); p = (float)z[i] / l;

m = log2(1 / p); k = m \* p;

h += k;

}

}

printf("\nEntropy = %f bits/symbol\n", h); R = r \* h;

printf("\nRate of Information = %f bits/sec\n", R);

return 0;

}

OUTPUT:

Enter the string: communication Length of string = 13.000000

|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | Frequency | Probability | Information |
| a | 1 | 0.076923 | 3.700440 bits |
| c | 2 | 0.153846 | 2.700440 bits |
| i | 2 | 0.153846 | 2.700440 bits |
| M | 2 | 0.153846 | 2.700440 bits |
| N | 2 | 0.153846 | 2.700440 bits |
| O | 2 | 0.153846 | 2.700440 bits |
| T | 1 | 0.076923 | 3.700440 bits |
| U | 1 | 0.076923 | 3.700440 bits |

Entropy = 2.931209 bits/symbol

Rate of Information = 0.000003 bits/sec

=== Code Execution Successful ===

# EXP 8 :- THE PROGRAM FOR IMPLEMENTION OF HUFF-MON CODING

# PROGRAM:

% Huffman Coding

%% Entering symbols and their probabilities symbol = input(‘Enter the symbol:- ’);

p = input(‘Enter the symbol probability:- ’); [dict,avglen] = huffmandict(symbol,p);

%% Printing the codeword code = dict;

for i=1:length(code)

code{i,2} = num2str(code{i,2}); end

disp(‘code=’); disp(code);

%% Printing entropy, efficiency, and redundancy [m,k] = size(p);

Hx = 0;

for c=1:k

hx = p(c)\*log2(1/p(c)); Hx = Hx+hx;

end

disp(‘Calculation of Entropy’); disp(Hx);

% calculation of coding efficiency disp(‘Calculation of coding efficiency’); n1 = (Hx/avglen);

n = n1\*100; disp(‘n=’); disp(n);

% calculation of redundancy disp(‘Calculation of redunduncy’); r = 1-n1;

disp(‘r=’); disp(r);

%% Printing encoded and decoded symbols

sig = input(‘Enter the symbols to be transmitted:- ’); disp(‘Symbols after encoding are: ’);

hcode = huffmanenco(sig,dict); disp(hcode);

disp(‘Encoded signals after decoding are: ’); dhsig = huffmandeco(hcode,dict); disp(dhsig);

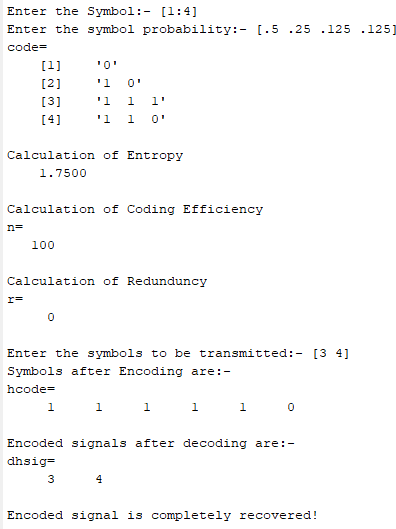
if(isequal(sig,dhsig))

disp(‘Encoded signal is completely recovered’);

else end

disp(‘Encoded signal is not properly recovered’);

# OUTPUT:



**EXP 8B :-THE PROGRAM FOR IMPLEMENTION OF SHANON-FANO CODING**

**PROGRAM:**

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

int arr[100][100], M;

float p[10], d[20];

void fano(int x, int g, int l) { float s1, s2, diff, min;

int s, e, i, j, k; s = x;

e = g;

if (s != e) {

// calculation of total sum s1 = 0;

for (i = s; i <= e; i++) { s1 = s1 + p[i];

}

// calculation of difference s2 = 0;

for (i = s; i < e; i++) { s2 = s2 + p[i];

diff = s2 - (s1 - s2); if (diff < 0)

diff = -diff; d[i] = diff;

}

// Calculation of minimum difference min = 999;

for (i = s; i <= (e - 1); i++) { if (min > d[i]) {

min = d[i];

k = i; // k is the index of minimum different element

}

}

// assign 0 and 1

for (j = s; j <= k; j++) arr[j][l] = 0;

for (j = k + 1; j <= e; j++) arr[j][l] = 1;

fano(s, k, l + 1);

fano(k + 1, e, l + 1);

}

}

int main() {

int i, j, Nk[10];

float temp, sum, a, E, H, Hi, Ni, N;

printf("\n Enter the number of messages: "); scanf("%d", &M);

sum = 0;

while (sum != 1) {

printf("\n Enter the probability of each message:\n "); for (i = 0; i < M; i++) {

scanf("%f", &a);

printf(" ");

if (a > 0 && a < 1) { p[i] = a;

} else {

printf("\n Re-enter the probability: "); i--;

}

}

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

printf("\n Probability of message is = %d = %.4f",i,p[i]);

}

sum = 0;

for (i = 0; i < M; i++) { sum = sum + p[i];

}

printf("\n\n The sum is: %.4f", sum); if (sum == 1)

break;

}

// Assigning constant value to array for (i = 0; i < M; i++) {

for (j = 0; j < M; j++) { arr[i][j] = 5;

}

}

printf("\n\n The entered probabilities are: ");

for (i = 0; i < M; i++) { printf("\n %.4f", p[i]);

}

// Arranging probabilities in descending order for (j = 0; j < M; j++) {

for (i = j + 1; i < M; i++) { if (p[j] < p[i]) {

temp = p[j]; p[j] = p[i]; p[i] = temp;

}

}

}

printf("\n\n The sorted probabilities are: "); for (i = 0; i < M; i++) {

printf("\n %.4f", p[i]);

}

fano(0, M - 1, 0);

printf("\n\n Probability\tCoding steps"); for (i = 0; i < M; i++) {

printf("\n %.4f\t\t\t", p[i]); for (j = 0; j < M; j++) {

if (arr[i][j] == 0 || arr[i][j] == 1)

printf("%d ", arr[i][j]);

}

}

// display the code and calculation of codeword length Nk printf("\n\n Code\t\t\tNk");

for (i = 0; i < M; i++) { Nk[i] = 0;

printf("\n");

for (j = 0; j < M; j++) { printf(" ");

if (arr[i][j] == 0 || arr[i][j] == 1) {

printf("%d", arr[i][j]);

Nk[i]++;

}

}

printf("\t\t\t%d", Nk[i]);

}

// Calculation of Entropy, average codeword length & efficiency N = 0;

H = 0;

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

Hi = p[i] \* (log(1 / p[i]) / log(2)); H = Hi + H;

Ni = (p[i] \* Nk[i]); N = Ni + N;

}

E = (H / N) \* 100;

printf("\n\n The entropy of the source is: H = %f", H); printf("\n The average codeword length is: N = %f", N); printf("\n The efficiency of the given code is: E = %f", E);

return 0;

}

# OUTPUT:

Enter the number of messages: 4

Enter the probability of each message:

.5

.25

.125

.125

Probability of message is = 0 = 0.5000 Probability of message is = 1 = 0.2500 Probability of message is = 2 = 0.1250 Probability of message is = 3 = 0.1250

The sum is: 1.0000

The entered probabilities are: 0.5000

0.2500

0.1250

0.1250

The sorted probabilities are:

0.5000

0.2500

0.1250

0.1250

Probability Coding steps 0.5000 0

0.2500 1 0

0.1250 1 1 0

0.1250 1 1 1

Code Nk

0 1

1 0 2

1 1 0 3

1 1 1 3

The entropy of the source is: H = 1.750000 The average codeword length is: N = 1.750000

The efficiency of the given code is: E = 100.000000

=== Code Execution Successful ===

Enter the probability of each message: 0.5000

0.0625

0.0625

0.0625

0.1250

0.1250

0.0312

0.0312

Probability of message is = 0 = 0.5000 Probability of message is = 1 = 0.0625 Probability of message is = 2 = 0.0625 Probability of message is = 3 = 0.0625 Probability of message is = 4 = 0.1250 Probability of message is = 5 = 0.1250 Probability of message is = 6 = 0.0312 Probability of message is = 7 = 0.0312

The sum is: 1.0000

The entered probabilities are:

0.5000

0.0625

0.0625

0.0625

0.1250

0.1250

0.0312

0.0312

The sorted probabilities are:

0.5000

0.1250

0.1250

0.0625

0.0625

0.0625

0.0312

0.0312

|  |  |
| --- | --- |
| Probability | Coding steps |
| 0.5000 | 0 |
| 0.1250 | 1 0 0 |
| 0.1250 | 1 0 1 |
| 0.0626 | 1 1 0 0 |
| 0.0625 | 1 1 0 1 |
| 0.0625 | 1 1 1 0 |
| 0.0312 | 1 1 1 1 0 |
| 0.0312 | 1 1 1 1 1 |
| Code | Nk |
| 0 | 1 |
| 1 0 0 | 3 |
| 1 0 1 | 3 |
| 1 1 0 0 | 4 |
| 1 1 0 1 | 4 |
| 1 1 1 0 | 4 |
| 1 1 1 1 0 | 5 |
| 1 1 1 1 1 | 5 |

The entropy of the source is: H = 2.312400 The average codeword length is: N = 2.312500

The efficiency of the given code is: E = 100.000000

# EXP 9 :-THE PROGRAM FOR IMPLEMENTION OF LINEAR BLOCK CODE

# PROGRAM:

clc; clear all; close all;

g = input('Enter the generator matrix: '); disp((g')');

disp('The order for linear block code for given generator matrix is: '); [n,k] = size(transpose(g));

disp('n ='); disp(n); disp('k ='); disp(k);

for i=1:2^k for j=k:-1:1

if rem(i-1,2^(-j+k+1))>=2^(-j+k) u(i,j) = 1;

else

u(i,j) = 0; end

end end

disp('The possible codewords are: '); C = rem((u\*g),2);

disp('C'); disp(C);

disp('The maximum hamming distance dmin for given block code is: '); d\_min = min(sum((C(2:2^k,:))'));

disp('d\_min ='); disp(d\_min);

%codeword

r = input('Enter the received codeword: '); disp('r =');

disp(r);

P = (g(:,n-k+2:n));

H = [transpose(P),eye(n-k)]; disp('PARITY CHECK MATRIX');

Ht = transpose(H); disp('Ht ='); disp(Ht);

disp('Syndrome of a given codeword is: '); S = rem(r\*Ht,2);

disp('S =');

disp(S);

for i=1:1:size(Ht) if(Ht(i,1:3)==S)

r(i) = 1-r(i); break;

end end

disp('The error is in bit: '); disp('i =');

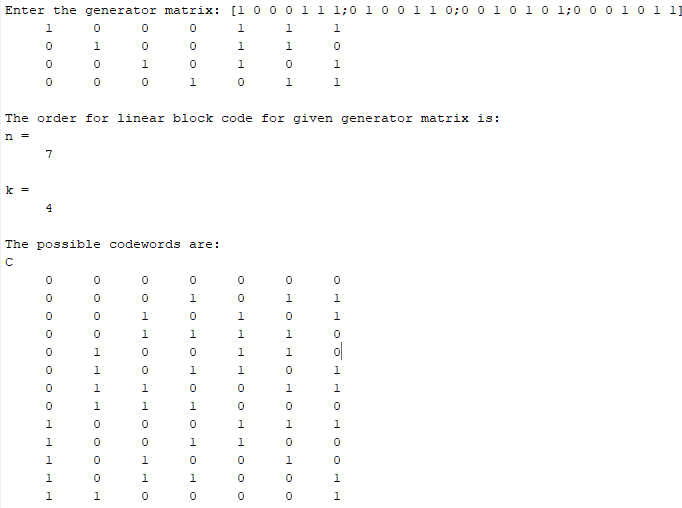
disp(i);

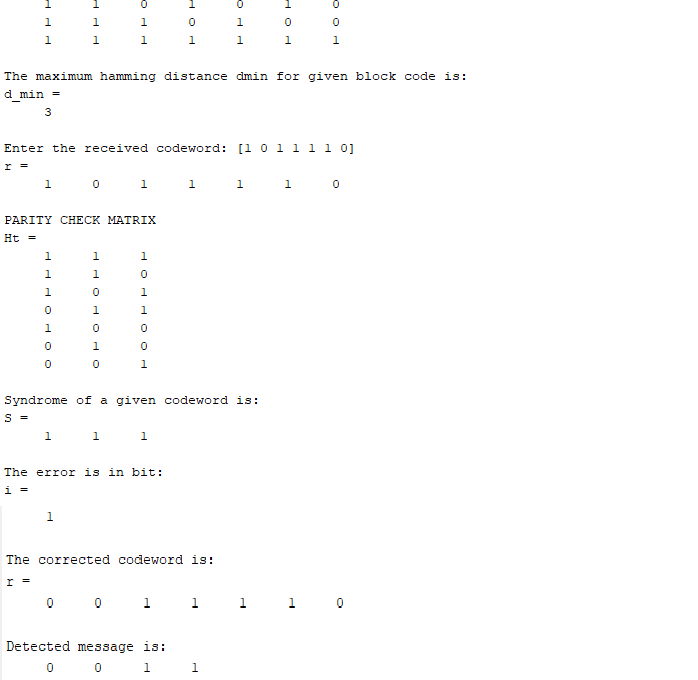
disp('The corrected codeword is: '); disp('r =');

disp(r);

disp('Detected message is: '); message = r(1,1:k); disp(message);

# OUTPUT:





**CYCLIC CODE NON-SYSTEMATIC FORM**

**PROGRAM 1:**

% Cyclic code non-systematic form clear all;

close all; clc;

m = 4;

mpoly = gf({ 1 1 1 0}, m);

gpoly = gf({1 0 1 1 }, m); cpoly = conv(mpoly, gpoly); disp(cpoly);

cpoly = gf(2^4) array. Primitive polynomial = D^4+D+1 (19 decimal)

# OUTPUT:

Array elements =

1 1 0 0 0 1 0

# CYCLIC SYSTEMATIC CODE

# PROGRAM 2:

% Cyclic systematic code clear all;

close all; clc;

n = input(‘Enter the length of codevector= ’); k = input(‘Enter the length of msg bits= ’);

g\_coeff = input(‘Enter the generator coefficient= ’); m = input(‘Enter the msg bits= ’);

y = [1];

a = zeros(1, n-k);

z1 = cat(2, y, a);

x = conv(z1, m);

x1 = abs(rem(x, 2));

[q,r] = deconv(x1, g\_coeff); r1 = abs(rem(r,2)); codeword = xor(x1,r1); disp(codeword);

# OUTPUT:

Enter the length of codevector= 7 Enter the length of msg bits= 4

Enter the generator coefficient= [1 0 1 1]

Enter the msg bits= [1 1 1 0]

Codeword=

1 1 1 0 1 0 0

# PROGRAM 3:

clc; close all; clear all;

n = input('Enter the length of Codeword: '); k = input('Enter the length of message: ');

gen\_coff = input('Enter the generator coefficient: '); rec = input('Enter the received codeword: ');

[q,r] = deconv(rec,gen\_coff); syn = abs(rem(r,2));

if syn==0 disp('No error');

else

disp('Error'); end

if syn==0

disp('No need of correction'); else

e = eye(n); for i=1:n

[x,y(i,:)] = deconv(e(i,:),gen\_coff); end

z = abs(rem(y,2)); for i=1:n

if syn==z(i,:) break;

end end disp('i ='); disp(i);

disp('Bit is in error'); corrected = xor(rec,e(i,:)); disp('corrected = '); disp(corrected);

end

# OUTPUT:

Enter the length of Codeword: 7 Enter the length of message: 4

Enter the generator coefficient: [1 0 1 1]

Enter the received codeword: [1 0 0 1 1 0 0]

Error i =

6

Bit is in error corrected =

1 0 0 1 1 1 0

# THIS CODES ARE ONLY PERFROMED IN MATLAB COMPILER

# clc;

# clear all;

# m=16;

# x=[0:m-1];

# n=1;

# off=0;

# z=pskmod(x,m);

# figure(1)

# scatterplot(z,n,off,"r+")

%constellation digrams od 16-psk

%in M-16 Symbols and in presnece of noise ,SNR=20 db,SNR=10 db

%16-PSK;

M=input('Number\_Symbols=');

SNR=input('SNR of QPSK system in dB=');

x1=randi([0 M-1],1000,1)

y1=pskmod(x1,M);

y1n=awgn(y1,SNR,'measured');

scatterplot(y1n);

y1r=pskdemod(y1n,M);

[num\_error,er\_rate]=symerr(x1,y1r)

