

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/313837510>

# Information Theory and Coding: Case Studies of Laboratory Experiments

**Method** · February 2017

DOI: 10.13140/RG.2.2.22158.77121

---

CITATIONS

0

---

READS

3,811

**1 author:**

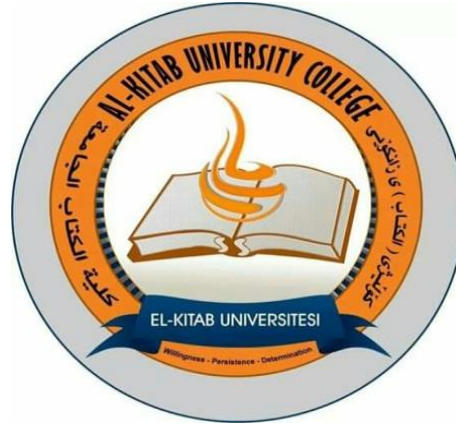


[Prof. Ayad Ghany Ismaeel](#)

Al-Kitab University

76 PUBLICATIONS 203 CITATIONS

[SEE PROFILE](#)



## **AL-KITAB UNIVERSITY COLLEGE**

**Department of Computer Techncl Engineering/  
Computer Communication and Network- Forth Stage**

# **Laboratory Manual of Information Theory and Coding**

**Prepared By  
Prof. Dr. Ayad Ghany Ismaeel**

**2016-2017**

## **INFORMATION THEORY AND CODING LABORATORY**

### **OBJECTIVES:**

*The student should be made to:*

- Be exposed to the information theories and their coding.
- Learn to implement the algorithms of coding.
- Learn to use and apply the information theory and coding algorithms.

### **LIST OF EXPERIMENTS:**

1. Review of Instructions, Statements, Tools and Functions of C/C++ Programming Languages:

- ❖ VARIABLE SCOPE,
- ❖ SWAPPING INTEGERS BY REFERENCE,
- ❖ CHECKING THE NUMBER EVEN OR ODD USING TERNARY OPERATOR,
- ❖ SORTING INTEGER NUMBERS,
- ❖ FACTORIAL USING RECURSION,
- ❖ FUNCTION OVERLOADING and
- ❖ INLINE FUNCTION

2. Implement the following coding algorithms:

- APPLIED THE ENCODING
- DISCRETE ENTROPY FOR PROBABILITY
- IMPLEMENT ENTROPY FOR PARTS OF MESSAGE
- COMPUTE THE ENTROPY OF MESSAGE/TEXT
- NOISELESS (NO NOISE) BINARY CHANNEL
- BINARY SYMMETRIC CHANNEL BSC CAPACITY
- SHANNON- FANO CODE ALGORITHM
- THE HUFFMAN- CODING ALGORITHM

### **PERIODS**

### **OUTCOMES:**

*At the end of the course, the student should be able to:*

- Define the information theories and the types of coding.
- Define the algorithms used in coding.
- Implement the information theories techniques.
- Compute the capacity of various types of channels.
- Develop the various coding algorithms.
- Use different OOP and open source tools (C++/C tutorials) for information theory and coding.

### **LIST OF EQUIPMENT FOR A BATCH OF 30 STUDENTS:**

**SOFTWARE:**

C / C++ or equivalent compiler like DEV C++.

**HARDWARE:**

Standalone desktops/Laptops 30 Nos.

.

## INDEX

<b>S.NO</b>	<b>EXPERIMENT NAME</b>	<b>Page No.</b>
1	1(a) VARIABLE SCOPE, 1(b) SWAPPING INTEGERS BY REFERENCE and 1(c) CHECKING THE NUMBER EVEN OR ODD USING TERNARY OPERATOR	5-8
2	2(a) SORTING INTEGER NUMBERS and 2(b) FACTORIAL USING RECURSION, 2(c) FUNCTION OVERLOADING and 2(d) INLINE FUNCTION	9-13
3	APPLIED THE ENCODING	14
4	DISCRETE ENTROPY FOR PROBABILITY	16
5	IMPLEMENT ENTROPY FOR PARTS OF MESSAGE	18
6	COMPUTE THE ENTROPY OF MESSAGE/TEXT	20
7	NOISELESS (NO NOISE) BINARY CHANNEL	22
8	BINARY SYMMETRIC CHANNEL BSC CAPACITY	24
9	BINARY SYMMETRIC CHANNEL CAPACITY: PRIVATE CASE	25
10	SHANNON- FANO CODE ALGORITHM	27
11	THE HUFFMAN- CODING ALGORITHM	30

Note: Each Experiment Lasts two/three weeks

**EX.No:1(a)**

**VARIABLE SCOPE**

**Aim:**

Write a c++ program illustrating variable scope.

**Algorithm:**

Step1: Use global variable as glo is 10

Step2: Use local variables as lo=20 , glo=40

Step3: Print valus of variables lo, glo (local variable)

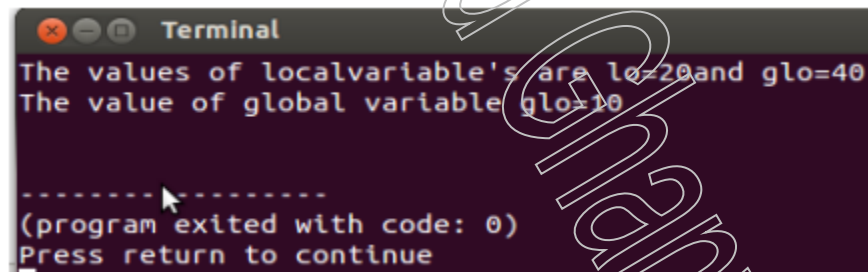
Step4: Print value of glo (global variable)

Step5: stop

**Program:**

```
#include<iostream>
using namespace std;
int glo=10;
main()
{
int lo=20,glo=40;
cout<<"The values of localvariable's are lo="<<lo<<"and glo="<<glo<<endl;
cout<<"The value of global variable glo="<<::glo<<endl;
}
```

**Output:**



```
Terminal
The values of localvariable's are lo=20and glo=40
The value of global variable glo=10
-----
(program exited with code: 0)
Press return to continue
```

**stdin:**

Standard input is empty

**stdout:**

The values of local variable's are lo= 20 and glo=40

The values of global variable glo=10

**RESULT:**

Thus the program was executed and verified successfully.

**EX.No.:1(b)**

**SWAPPING INTEGERS BY REFERENCE**

**Aim:**

**Write a C++ program illustrating Swapping integer values by reference.**

**Algorithm:**

**Step1: Start**

**Step2: Use a=100, b=200**

**Step3: Print ' before swapping, values of a, b are „**

**Step4: Call function swap (a, b)**

**Step5: Print „After swapping, values of a, b are“**

**Step6: stop**

**Algorithm for function swap(&x,&y)**

**Step1: Use variable temp.**

**Step2: temp = x;**

**Step3: x = y;**

**Step4: y = temp;**

**Program:**

```
#include <iostream>
using namespace std;
void swap(int &x, int &y); // function declaration
int main ()
{ // local variable declaration
  int a = 100;
  int b = 200;
  cout << "Before swap, value of a :" << a << endl;
  cout << "Before swap, value of b :" << b << endl;
  /* calling a function to swap the values using variable reference.*/
  swap(a, b);
  cout << "After swap, value of a :" << a << endl;
  cout << "After swap, value of b :" << b << endl;
  return 0;
}
// function definition to swap the values.
void swap(int &x, int &y)
{
  int temp;
  temp = x; /* save the value at address x in temp */
  x = y; /* put y into x */
  y = temp; /* put temp into y */
}
```

**stdin:**

Standard input is empty

**stdout:**

Executing the program....

\$demo

Before swap, value of a :100

Before swap, value of b :200  
After swap, value of a :200  
After swap, value of b :100

**RESULT:**

Thus the program was executed and verified successfully.



**EX.No.:1(c) CHECKING THE NUMBER EVEN OR ODD USING TERNARY OPERATOR**

**Aim:**

Write a C++ program to illustrate checking whether the no is even or odd using ternary operator

**Algorithm:**

Step1: start

Step2: print 'enter a number'

Step3: read n value

Step4: if  $n \% 2 == 0$  (use Ternary Operator)

4.1: print the number is even

Else

4.2: print the number is odd

Step5: stop.

**Program:**

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
int n;
```

```
cout<<"Enter a number"<<endl;
```

```
cin>>n;
```

```
((n%2)==0)?cout<<"The number "<<n<<" is Even"<<endl:cout<<"The number "<<n<<" is  
odd"<<endl;
```

```
return 0;
```

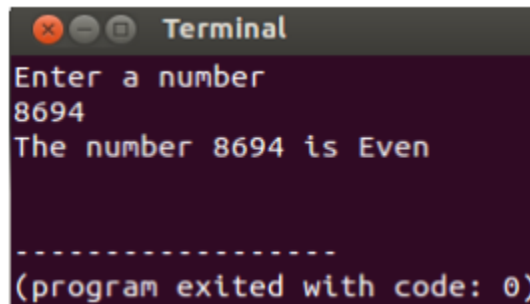
```
}
```

**stdin:**

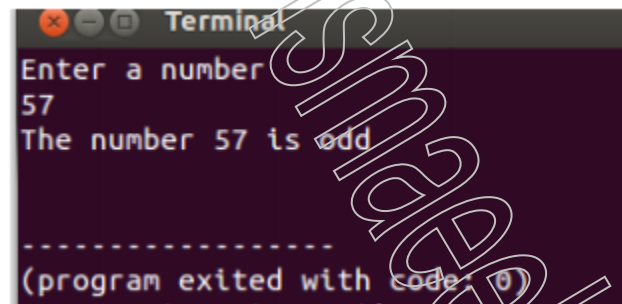
Standard input is empty

**stdout:**

Implement two time one of them even number and another one by odd number.



```
Terminal
Enter a number
8694
The number 8694 is Even
-----
(program exited with code: 0)
```



```
Terminal
Enter a number
57
The number 57 is odd
-----
(program exited with code: 0)
```

**RESULT:**

Thus the program was executed and verified successfully.

EX.No.:2(a)

## SORTING INTEGER NUMBERS

**Aim:****Write a c++ program illustrating to sort integer numbers.****Algorithm:****step1: start.****step2: Take an array X[10]****step3: Print „enter array size“****step4: read n value****step5: print 'enter elements in to array'****step6: for i=0 to n-1 insteps of 1 repeat step7****step7: read X[i]****[end for]****step8: call function insertion\_sort(X,n)****step9: print 'After sorting, the Array elements are...'****step10: for i=0 to n-1 insteps of 1 repeat step11****step11: print X[i]****[end for]****step12: stop.****Algorithm for function insertion\_sort(X,n)****Step1: for i=1 to n-1 insteps of 1 repeat step7****1.1: key=X[i] , pos=i.****1.2: repeat until pos>0 &&X[pos-1]>key****1.2.1:X[pos]=X[pos-1]****1.2.2: pos=pos-1****1.2.3: X[pos]=key;****[end while]****[end for]****Program:**

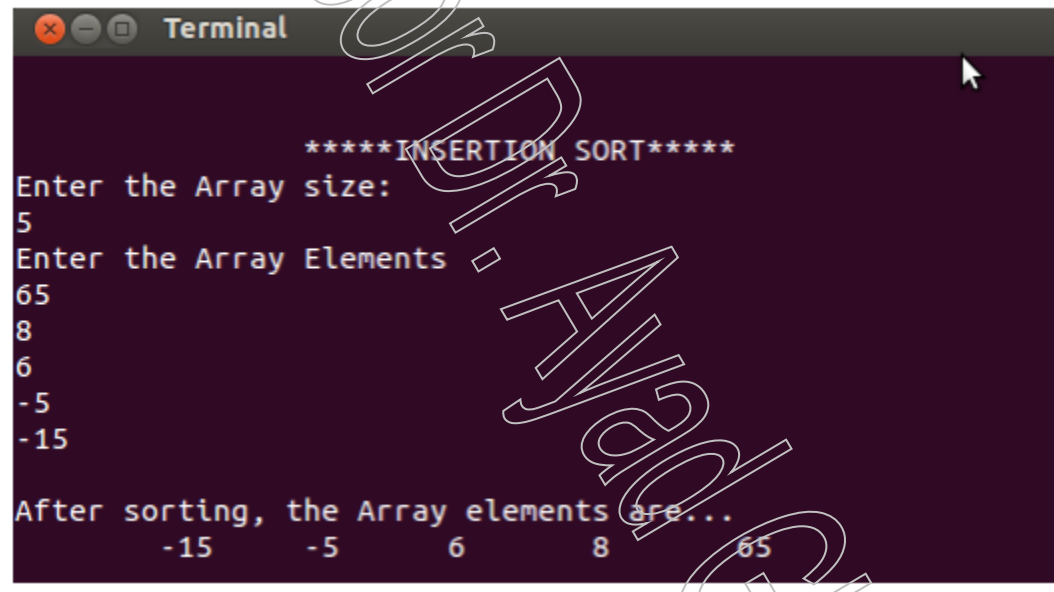
```

#include <cstdlib>
#include <iostream>
#define MAX 20
using namespace std;
void insertion_sort(int X[], int n);
int main()
{
    int array[MAX], n;
    cout<<"\n\n\t*****INSERTION SORT*****"<<endl;
    cout<<"Enter the Array size:"<<endl;
    cin>>n;
    cout<<"Enter the Array Elements"<<endl;
    for(int i=0;i<n;i++)
        cin>>array[i];
    insertion_sort(array,n);
    cout<<"\nAfter sorting, the Array elements are...\n";
    for(int i=0;i<n;i++)
        cout<<"\t"<<array[i];
    cout<<endl;
    return 0;
} //end of main
void insertion_sort(int X[],int n)
{
    int key,i,pos;
    for(i=1;i<n;i++)
    {

```

```
key=X[i];
pos=i;
while((pos>0)&&(X[pos-1]>key))
{
X[pos]=X[pos-1];
pos=pos-1;
X[pos]=key;
} //end of while loop
//end of for loop
//end of insertion_sort
```

**Output:**



```
*****INSERTION SORT*****
Enter the Array size:
5
Enter the Array Elements
65
8
6
-5
-15

After sorting, the Array elements are...
-15    -5     6     8     65
```

**RESULT:**

Thus the program was executed and verified successfully.

EX.No.:2(b)

## FACTORIAL USING RECURSION

**Aim:**

Write a C++ program illustrating factorial using recursion.

**Algorithm:**

step1: start

step2: print „enter a number“

step3: read n value

step4: call function factorial(n) and assign return value to k

step5: Print ' Factorial of „ n „ is „ k

step6: stop

**Algorithm for function factorial (n)**

step1: if n==0

then return 1.

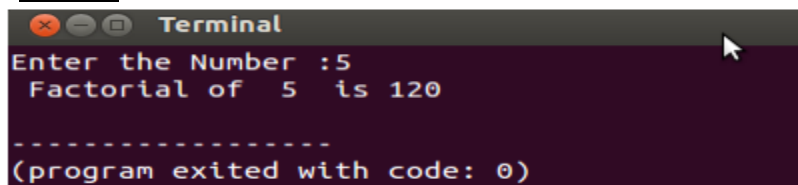
step2: else

return(n \* factorial(n-1))

**Program:**

```
#include<iostream>
using namespace std;
long factorial(int); //Function declaration
int main()
{
    int n;
    long int k; // Variable Declaration
    cout<<"Enter the Number :";
    cin>>n; // Get Input Value
    k=factorial(n); // Factorial Function Call
    cout<<" Factorial of "<<n<<" is "<<k;
    return 0;
}
// Factorial Function using recursion
long int factorial(int n)
{
    if (n == 0)
        return 1;
    else
        return(n * factorial(n-1));
}
```

**Output:**



```
Terminal
Enter the Number :5
Factorial of 5 is 120
-----
(program exited with code: 0)
```

**RESULT:**

Thus the program was executed and verified successfully.

**EX.No.:2(c)****FUNCTION OVERLOADING****Aim:**

Write a C++ program illustrating function overloading.

**Algorithm:**

**Step1:** Declare long add(long, long);

**Step2:** Declare float add(float, float);

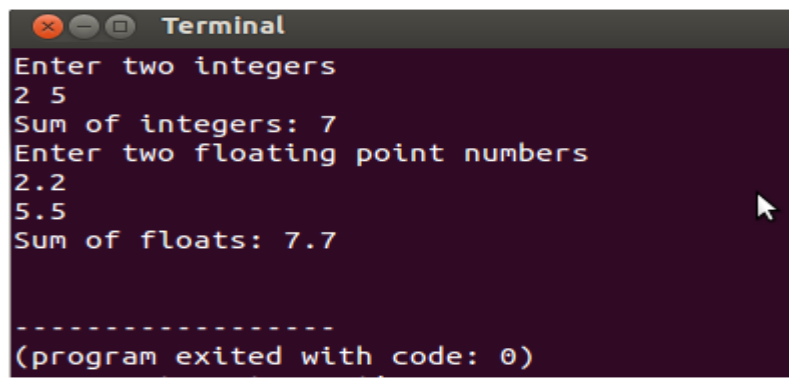
**Step3:** Call these functions separately in main

**Step4:** Define these functions separately.

**Step5:** stop

**Program:**

```
#include <iostream>
using namespace std;
/* Function arguments are of different data type */
long add(long, long);
float add(float, float);
int main()
{
    long a, b, x;
    float c, d, y;
    cout << "Enter two integers\n";
    cin >> a >> b;
    x = add(a, b);
    cout << "Sum of integers: " << x << endl;
    cout << "Enter two floating point numbers\n";
    cin >> c >> d;
    y = add(c, d);
    cout << "Sum of floats: " << y << endl;
    return 0;
}
```

**Output:**


```
Terminal
Enter two integers
2 5
Sum of integers: 7
Enter two floating point numbers
2.2
5.5
Sum of floats: 7.7

-----
(program exited with code: 0)
```

**RESULT:**

Thus the program was executed and verified successfully.

EX.No.:2(d)

## INLINE FUNCTION

**Aim:**

Write a C++ program illustrating inline functions.

**Algorithm:**

Step1: start

Step2: read a,b

Step3: call an inline function mul() to multiply and return the value of a\*b

Step4: call an inline function div() to divide and return the value of a/b

Step5: stop

**Program:**

```
#include <iostream>
```

```
using namespace std;
```

```
inline int mul(int x, int y)
```

```
{
```

```
return x*y;
```

```
}
```

```
inline float div(int x, int y)
```

```
{
```

```
return (float)x/y;
```

```
}
```

```
int main()
```

```
{
```

```
int a, b, product;
```

```
float division;
```

```
cout << "Enter two integers\n";
```

```
cin >> a >> b;
```

```
product = mul(a, b);
```

```
cout << "Product of integers: " << product << endl;
```

```
division = div(a,b);
```

```
Object-Oriented Programming Lab Computer Science Engineering
```

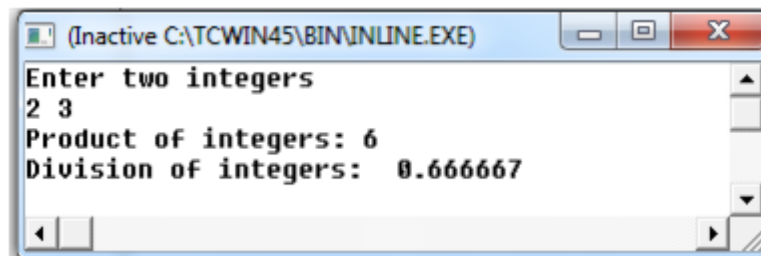
```
RAGHU INSTITUTE OF TECHNOLOGY 40
```

```
cout << "Division of integers: " << division << endl;
```

```
return 0;
```

```
}
```

**Output:**



**RESULT:**

Thus the program was executed and verified successfully.

**EX.No.:3**

## **APPLIED THE ENCODING**

### **AIM:**

Develop a program to implement The algorithm of Encoding of messages.

### **ALGORITHM**

When a sequence a series on n successive tokens appears (Message):

Step 1: Replace series with a token and a count number of occurrences.

Step 2: Usually need to have a special flag

Step 3: Denote when the repeated token appears:

- \* Put the flag.
- \* Put many the symbol is repeted and write it
- \* then write this symbol
- \* Go to find another repetition symbol
  - If there is return to step 3.
  - Else Stop.

### **PROGRAM :**

```
#include <iostream>
#include <string.h>
using namespace std;
int main() {
    int i,j,cnt,l,count[50]={0};
    char str[50]="sleepzzzzzzzzzzzzzzzzzz";
    printf("\n\tOriginal String is: %s",str);
    printf("\n\n\tEncoded String is: ");
    l = strlen(str);
    for(i=0;i<l;i*=1) {
        j = 0;
        count[i] = 1;
        do {
            j++;
            if(str[i+j] == str[i])
                count[i]++;
        } while(str[i+j]==str[i]);
        if(count[i]==1)
            printf("%c",str[i++]);
        else {
            printf("$%d%c",count[i],str[i]);
            i += count[i];
        }
    }
    getchar();
}
```

**stdin:**

Standard input is empty

**stdout:**



```
C:\Program Files\Dev-Cpp\Encoded.exe

Original String is: sleepzzzzzzzzzzzzzzzzzzzzzz
Encoded String is: s!$2ep$19z_
```

**RESULT:**

Thus the program was executed and verified successfully.



**EX.No.: 4      DISCRETE ENTROPY FOR PROBABILITY**

**AIM:**

Develop a program to Compute the Entropy in case of Discrete Algorithm.

**ALGORITHM DESCRIPTION:**

Find the Entropy for each following probability, and

- The 0.1 Probability
- The 0.15 Probability
- The 0.2 Probability
- The 0.25 Probability
- The 0.3 Probability

Then the Entropy of all message.

**PROGRAM:**

```
#include <iostream>
#include <cmath>
using namespace std;
int main(){
    int i;
    float p,sum=0, it;
    p=0.05;
    for(i=1;i<=5;i++) {
        p=0.05+p;
        it=p*(log(1/p))/log(2);
        sum=sum+it;
        cout<<"probability="<<p<<"    Entropy="<<it<<endl;
    }
    cout<<"The all Entropy="<<sum<<endl;
}
```

**stdin:**

Standard input is empty

**stdout:**

```
sh-4.3$ main
probability=0.1 Entropy=0.332193
probability=0.15 Entropy=0.410545
probability=0.2 Entropy=0.464386
probability=0.25 Entropy=0.5
probability=0.3 Entropy=0.52109
The all Entropy=2.22821
```

**RESULT:**

Thus the program was executed and verified successfully.

**EX.No.: 5**

**IMPLEMENT ENTROPY FOR PARTS OF MESSAGE**

**AIM:**

Develop a program to Compute Entropy of 4 Parts of Message

**ALGORITHM DESCRIPTION:**

Suppose you have 4 messages each of them has the following probability:

Enter The 1 Probability : 0.1

Enter The 2 Probability : 0.2

Enter The 3 Probability : 0.3

Enter The 4 Probability : 0.4

Find the Information Content of these Messages

**PROGRAM :**

```
#include <iostream>
#include <cmath>
using namespace std;
int main(){
    int i;
    float p,sum=0, it;
    p=0.0;
    for(i=1;i<=4;i++) {
        p=0.1+p;
        it=p*(log(1/p))/log(2);
        sum=sum+it;

    cout<<"probability="<<p<<
    "  Entropy="<<it<<endl;
    }
    cout<<"The all
Entropy="<<sum<<endl;
}
```

**Stdin:**

The Input is Empty

**OUTPUT:**

Executing the program in case of  $(it=p*(\log(1/p))/\log(2);)$

\$demo

probability=0.1 Entropy=0.332193

probability=0.2 Entropy=0.464386

probability=0.3 Entropy=0.52109

probability=0.4 Entropy=0.528771

The all Entropy=1.84644

### **RESULT:**

Thus the program was executed and verified successfully.

**EX.No.: 6****COMPUTE THE ENTROPY OF MESSAGE/TEXT****AIM:**

To write a program to Find the Entropy of certain message.in C++

**DESCRIPTION ALGORITHM**

- Entropy is the expected value of the measure of information content in a system. In general, the Shannon entropy of a variable X is defined as:

$$H(X) = \sum_{x \in \Omega} P(x) I(x)$$

- Where the information content  $I(x) = -\log_b P(x)$ . If the base of the logarithm  $b = 2$ , the result is expressed in bits, a unit of information. Therefore, given a string S of length n where  $P(s_i)$  is the relative frequency of each character, the entropy of a string in bits is:

$$H(S) = -\sum_{i=0}^n P(s_i) \log_2(P(s_i))$$

- For this task, use "1223334444" as an example. The result should be around 1.84644 bits.

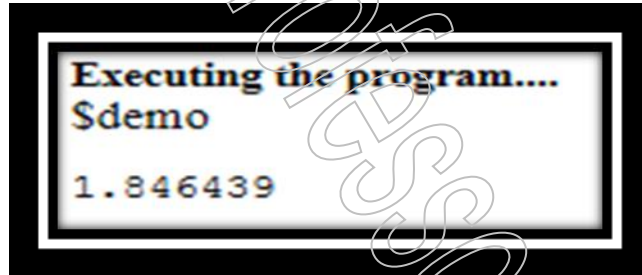
**PROGRAM :**

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;
double calcEntropy(string str) {
    int freqs[256]={0};
    int i;
    double entropy=0.0;
    double size=str.size();
    for(i=0;i<str.size();i++) freqs[str[i]]++;
    for(i=0;i<256;i++) {
        if(freqs[i]!=0) entropy+=(freqs[i]/size)*log2(freqs[i]/size);
    }
    entropy=-entropy;
    cout<<"String = "<<str<<"\n";
    cout<<"Entropy = "<<entropy<<"\n";
    cout<<"\n";
    return entropy;
}
int main() {
    string inputs[]={
        "1223334444"
    };
    for(int i=0;i<sizeof(inputs)/sizeof(string);i++) {
        calcEntropy(inputs[i]);
    }
    return 0;
}
```

**stdin:**

Standard input is empty

**stdout:**



**RESULT:**

Thus the program was executed and verified successfully.

**EX.No.: 7 NOISELESS (NO NOISE) BINARY CHANNEL****AIM:**

Develop and Implement Program to Compute the Capacity of Noiseless Binary Channel.

**DESCRIPTION :**

Consider a binary symmetric communication channel, whose input source is the alphabet  $X = \{0, 1\}$  with probabilities  $\{0.5, 0.5\}$ ; whose output alphabet is  $Y = \{0, 1\}$ ; and whose channel matrix is:

$$\begin{bmatrix} 1-e & e \\ e & 1-e \end{bmatrix}$$

- Where  $q$  is the probability of transmission error:
- What is the entropy of the source,  $H(X)$ ?
- What is the probability distribution of the outputs,  $p(Y)$ , and the entropy of this output distribution,  $H(Y)$ ?
- What is the joint probability distribution for the source and the output,  $p(X, Y)$ , and what is the joint entropy,  $H(X, Y)$ ?
- What is the mutual information of this channel,  $I(X; Y)$ ?
- How many values are there for  $q$  for which the mutual information of this channel is maximal? What are those values, and what then is the capacity of such a channel in bits?
- For what value of  $q$  is the capacity of this channel minimal? What is the channel capacity in that case?.

Solving the Problem:

- Entropy of the source,  $H(X)$ , is 1 bit.
- Output probabilities are  $p(y=0) = (0.5)(1-e) + (0.5)e = 0.5$  and  $p(y=1) = (0.5)(1-e) + (0.5)e = 0.5$ . Entropy of this distribution is  $H(Y) = 1$  bit, just as for the entropy  $H(X)$  of the input distribution.
- Joint probability distribution  $p(X, Y)$  is:

$$\begin{bmatrix} 0.5(1-e) & 0.5e \\ 0.5e & 0.5(1-e) \end{bmatrix}$$

and the entropy of this joint distribution is  $H(X, Y) = -\sum_{x,y} p(x, y) \log_2 p(x, y)$   
 $= -(1-e) \log(0.5(1-e)) - e \log(0.5e) = (1-e) - (1-e) \log(1-e) + e - e \log(e)$   
 $= 1 - e \log(e) - (1-e) \log(1-e)$

- The mutual information is  $I(X; Y) = H(X) + H(Y) - H(X, Y)$ , which we can evaluate from the quantities above as:  $1 + e \log(e) + (1-e) \log(1-e)$ .
- In the two cases of  $e=0$  and  $e=1$  (perfect transmission, and perfectly erroneous transmission), the mutual information reaches its maximum of 1 bit and this is also then the channel capacity.
- If  $e=0.5$ , the channel capacity is minimal and equal to 0.

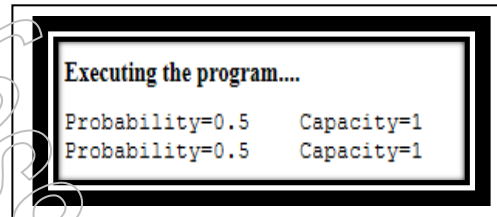
**PROGRAM:**

```
#include <iostream>
#include <cmath>
using namespace std;
int main(){
    int i;
    float p, it;
    p=0.5;
    for(i=1;i<=2;i++) { // 1st if 1 then 0, 2nd if 0 then 1
        it=p*(log(1/p))/log(2)+(1-p)*(log(1/(1-p)))/log(2);
        cout<<"Probability="<<p<<" Capacity="<<it<<endl;
    }
}
```

**Stdin:**

The input is Empty

**Stdout:**

A screenshot of a terminal window with a black border. The text inside is as follows:

```
Executing the program....  
Probability=0.5    Capacity=1  
Probability=0.5    Capacity=1
```

**RESULT:**

Thus the program was executed and verified successfully.



**EX.No.: 8****BINARY SYMMETRIC CHANNEL BSC CAPACITY****AIM:**

Can computing Binary Entropy Function (Channel Capacity) as follow:

$$C = 1 - H(p)$$

Write Program for BSC when  $p=0.1$  find the  $H_p = 0.468 \approx 0.47$  and Capacity =  $0.53 \approx 0.531$ .

**ALGORITHM:**

1. Can computing Binary Entropy Function (Channel Capacity) as follow:

$$H_2(p) \equiv H(p, 1 - p)$$

2. then we can find:

$$H_2(p) \equiv H(p, 1 - p) = p \log \frac{1}{p} + (1 - p) \log \frac{1}{(1-p)}.$$

3. and we have:

$$P = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$$

4. Binary Symmetric Channel Capacity is:

$$C = 1 - H(p)$$

**PROGRAM**

```
#include <iostream>
#include <cmath>
using namespace std;
int main(){
    float p, Hp, C;
    p=0.1;
    // 1st if 1 then 0, 2nd if 0 then 1
    Hp=p*(log(1/p))/log(2)+(1-p)*(log(1/(1-p)))/log(2);
    C=1-Hp;
    cout<<"Probability of p="<<p<<" Binary Entropy Function of Hp="<<Hp<<endl;
    cout<<"1-Hp -->Capacity="<<C<<endl;
}
```

**Stdin:**

The input is empty

**Stdout:**

Executing the program....

\$demo

Probability of p=0.1 Binary Entropy Function of Hp=0.468996  
1-Hp -->Capacity=0.531004

**RESULT:**

Thus the program was executed and verified successfully.

**EX.No.: 9 BINARY SYMMETRIC CHANNEL CAPACITY: PRIVATE CASE****AIM:**

Can Computing BSC (Channel Capacity) in Private Case Study As Follow:

$$I(X;Y) = H(Y) - H(Y|X)$$

Write Program For BSC of Private Case Study To Compute Capacity.

**DESCRIPTION:**

Consider the binary symmetric channel again, with  $f = 0.15$  and  $P_X : \{p_0 = 0.9, p_1 = 0.1\}$ . We already evaluated the marginal probabilities  $P(y)$  implicitly above:  $P(y=0) = 0.78$ ;  $P(y=1) = 0.22$ . The mutual information is:

$$I(X;Y) = H(Y) - H(Y|X).$$

What is  $H(Y|X)$ ? It is defined to be the weighted sum over  $x$  of  $H(Y|x)$ ; but  $H(Y|x)$  is the same for each value of  $x$ :  $H(Y|x=0)$  is  $H_2(0.15)$ , and  $H(Y|x=1)$  is  $H_2(0.15)$ . So

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) \\ &= H_2(0.22) - H_2(0.15) \\ &= 0.76 - 0.61 = 0.15 \text{ bits.} \end{aligned}$$

This may be contrasted with the entropy of the source  $H(X) = H_2(0.1) = 0.47$  bits.

Note: here we have used the binary entropy function  $H_2(p) \equiv H(p, 1-p) = p \log \frac{1}{p} + (1-p) \log \frac{1}{(1-p)}$ .

**PROGRAM:**

```
#include <iostream>
#include <cmath>
using namespace std;
int main()

{
    float py, pyx, Hy, Hyx, C;
    py=0.22;
    pyx=0.15;
    // 1st if 1 then 0, 2nd if 0 then 1
    Hy=py*(log(1/py))/log(2)+(1-py)*(log(1/(1-py)))/log(2);
    Hyx=pyx*(log(1/pyx))/log(2)+(1-pyx)*(log(1/(1-pyx)))/log(2);
    C=Hy-Hyx;
    cout<<"Probability of PY="<<py<<" Probability of PYx="<<pyx<<"Capacity="<<C<<endl;
    system("pause");
}
```

**Stdin:**

The input is Empty

**Stdout:**

```
Executing the program....  
Probability of PY=0.22    Probability of PYx=0.15    Capacity=0.150327
```

**RESULT:**

Thus the program was executed and verified successfully.

**EX.No.: 10****SHANNON- FANO CODE ALGORITHM****AIM:**

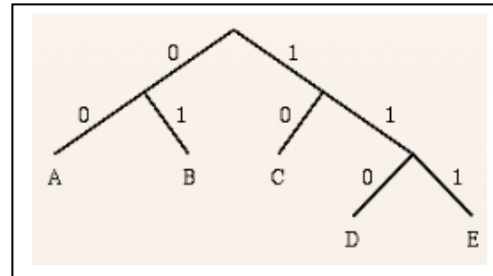
This is a basic information theoretic algorithm. A simple example will be used to illustrate the Shannon Fano algorithm:

**DESCRIPTION:**

Symbol A B C D E

Count 15 7 6 6 5

Encoding for the Shannon-Fano Algorithm:



A top-down approach

1. Sort symbols according to their frequencies/probabilities, e.g., ABCDE.
2. Recursively divide into two parts, each with approx. same number of counts.

Symbol Count  $\log(1/p)$  Code Subtotal (# of bits)

A	15	1.38	00	30
B	7	2.48	01	14
C	6	2.70	10	12
D	6	2.70	110	18
E	5	2.96	111	15

TOTAL (# of bits): 89

**PROGRAM:**

```

#include<stdio.h>
#include<conio.h>
#include<string.h>
struct node
{
char sym[10];
float pro;
int arr[20];
int top;
}s[20];

typedef struct node node;

void prints(int l,int h,node s[])
{
int i;
for(i=l;i<=h;i++)
printf("\n%s\t%f",s[i].sym,s[i].pro);
}

void shannon(int l,int h,node s[])
{
float pack1=0,pack2=0,diff1=0,diff2=0;
int i,d,k,j;
if((l+1)==h || l==h || l>h)
{
if(l==h || l>h)
return;

```

```

s[h].arr[++(s[h].top)]=0;
s[l].arr[++(s[l].top)]=1;
return;
}
else
{
for(i=1;i<=h-1;i++)
pack1=pack1+s[i].pro;
pack2=pack2+s[h].pro;
diff1=pack1-pack2;
if(diff1<0)
diff1=diff1*-1;
j=2;
while(j!=h-1+1)
{
k=h-j;
pack1=pack2=0;
for(i=1;i<=k;i++)
pack1=pack1+s[i].pro;
for(i=h;i>k;i--)
pack2=pack2+s[i].pro;
diff2=pack1-pack2;
if(diff2<0)
diff2=diff2*-1;
if(diff2>=diff1)
break;
diff1=diff2;
j++;
}
k++;
for(i=1;i<=k;i++)
s[i].arr[++(s[i].top)]=1;
for(i=k+1;i<=h;i++)
s[i].arr[++(s[i].top)]=0;
shannon(1,k,s);
shannon(k+1,h,s);
}

int main()
{
int n,i,j;
float x,total=0;
char ch[10];
node temp;
printf("Enter How Many Symbols Do You Want To Enter\t: ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter symbol %d ---> ",i+1);
scanf("%s",ch);
strcpy(s[i].sym,ch);
}
for(i=0;i<n;i++)
{
printf("\n\tEnter probability for %s ---> ",s[i].sym);
scanf("%f",&x);
s[i].pro=x;
total=total+s[i].pro;
if(total>1)
{
printf("\n\tThis probability is not possible.Enter new probability");
total=total-s[i].pro;
i--;
}
}
s[i].pro=1-total;
for(j=1;j<=n-1;j++)

```

```

{
for(i=0;i< n-1;i++)
if((s[i].pro)>(s[i+1].pro))
{
temp.pro=s[i].pro;
strcpy(temp.sym,s[i].sym);
s[i].pro=s[i+1].pro;
strcpy(s[i].sym,s[i+1].sym);
s[i+1].pro=temp.pro;
strcpy(s[i+1].sym,temp.sym);
}

for(i=0;i< n;i++)
s[i].top=-1;

shannon(0,n-1,s);
printf("-----");
printf("\n\n\tSymbol\tProbability\tCode");
for(i=n-1;i>=0;i--)
{
printf("\n\t%s\t%f\t",s[i].sym,s[i].pro);
for(j=0;j<=s[i].top;j++)
printf("%d",s[i].arr[j]);
}
printf("\n-----");
getch();
return 0;
}

```

**Stdin:**

Enter How Many Symbols Do You Want To Enter : 6

Enter symbol 1 ---> a

Enter symbol 2 ---> b

Enter symbol 3 ---> c

Enter symbol 4 ---> d

Enter symbol 5 ---> e

Enter symbol 6 ---> f

Enter probability for a ---> 0.3

Enter probability for b ---> 0.25

Enter probability for c ---> 0.20

Enter probability for d ---> 0.12

Enter probability for e ---> 0.08

Enter probability for f ---> 0.05

**Stdout:**

Symbol Probability Code

a	0.300000	00
b	0.250000	01
c	0.200000	10
d	0.120000	110
e	0.080000	1110
f	0.050000	1111

**RESULT:**

Thus the program was executed and verified successfully.

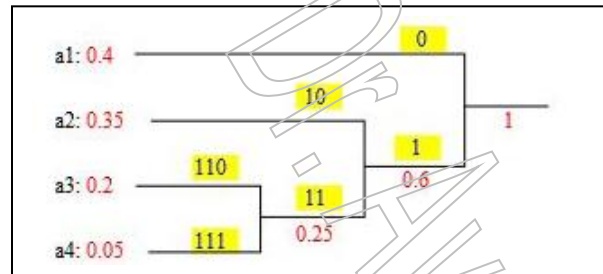
**EX.No.: 11 THE HUFFMAN- CODING ALGORITHM****AIM:**

This is a basic information theoretic algorithm. A simple example will be programmed in C++ for Huffman Coding algorithm:

**DESCRIPTION:**

The Huffman coding scheme takes each symbol and its weight (or frequency of occurrence), and generates proper encodings for each symbol taking account of the weights of each symbol, so that higher weighted symbols have fewer bits in their encoding. (See the WP article for more information).

A Huffman encoding can be computed by first creating a tree of nodes:



Huffman coding example.jpg

1. Create a leaf node for each symbol and add it to the priority queue.
  2. While there is more than one node in the queue:
    - Remove the node of highest priority (lowest probability) twice to get two nodes.
    - Create a new internal node with these two nodes as children and with probability equal to the sum of the two nodes' probabilities.
    - Add the new node to the queue.
  3. The remaining node is the root node and the tree is complete.
- Traverse the constructed binary tree from root to leaves assigning and accumulating a '0' for one branch and a '1' for the other at each node. The accumulated zeros and ones at each leaf constitute a Huffman encoding for those symbols and weights:

Using the characters and their frequency from the string "this is an example for huffman encoding", create a program to generate a Huffman encoding for each character as a table.

**PROGRAM:**

```

#include <iostream>
#include <queue>
#include <map>
#include <climits> // for CHAR_BIT
#include <iterator>
#include <algorithm>

const int UniqueSymbols = 1 << CHAR_BIT;
const char* SampleString = "this is an example for huffman encoding";

typedef std::vector<bool> HuffCode;
typedef std::map<char, HuffCode> HuffCodeMap;

class INode
{
public:
    const int f;

    virtual ~INode() {}
  
```



```

protected:
    INode(int f) : f(f) {}
};

class InternalNode : public INode
{
public:
    INode *const left;
    INode *const right;

    InternalNode(INode* c0, INode* c1) : INode(c0->f + c1->f), left(c0), right(c1) {}
    ~InternalNode()
    {
        delete left;
        delete right;
    }
};

class LeafNode : public INode
{
public:
    const char c;

    LeafNode(int f, char c) : INode(f), c(c) {}
};

struct NodeCmp
{
    bool operator()(const INode* lhs, const INode* rhs) const { return lhs->f > rhs->f; }
};

INode* BuildTree(const int (&frequencies)[UniqueSymbols])
{
    std::priority_queue<INode*, std::vector<INode*>, NodeCmp> trees;

    for (int i = 0; i < UniqueSymbols; ++i)
    {
        if(frequencies[i] != 0)
            trees.push(new LeafNode(frequencies[i], (char)i));
    }
    while (trees.size() > 1)
    {
        INode* childR = trees.top();
        trees.pop();

        INode* childL = trees.top();
        trees.pop();

        INode* parent = new InternalNode(childR, childL);
        trees.push(parent);
    }
    return trees.top();
}

void GenerateCodes(const INode* node, const HuffCode& prefix, HuffCodeMap& outCodes)
{
    if (const LeafNode* lf = dynamic_cast<const LeafNode*>(node))
    {
        outCodes[lf->c] = prefix;
    }
    else if (const InternalNode* in = dynamic_cast<const InternalNode*>(node))
    {
        HuffCode leftPrefix = prefix;
        leftPrefix.push_back(false);
        GenerateCodes(in->left, leftPrefix, outCodes);

        HuffCode rightPrefix = prefix;
        rightPrefix.push_back(true);
    }
}

```



```

        GenerateCodes(in->right, rightPrefix, outCodes);
    }
}

int main()
{
    // Build frequency table
    int frequencies[UniqueSymbols] = {0};
    const char* ptr = SampleString;
    while (*ptr != '\0')
        ++frequencies[*ptr++];

    INode* root = BuildTree(frequencies);

    HuffCodeMap codes;
    GenerateCodes(root, HuffCode(), codes);
    delete root;

    for (HuffCodeMap::const_iterator it = codes.begin(); it != codes.end(); ++it)
    {
        std::cout << it->first << " ";
        std::copy(it->second.begin(), it->second.end(),
            std::ostream_iterator<bool>(std::cout));
        std::cout << std::endl;
    }
    return 0;
}

```

**Stdin:**

The Input is Empty

**Stdout:**

```

110
a 1001
c 101010
d 10001
e 1111
f 1011
g 101011
h 0101
i 1110
l 01110
m 0011
n 000
o 0010
p 01000
r 01001
s 0110
t 01111
u 10100
x 10000

```

**RESULT:**

Thus the program was executed and verified successfully.