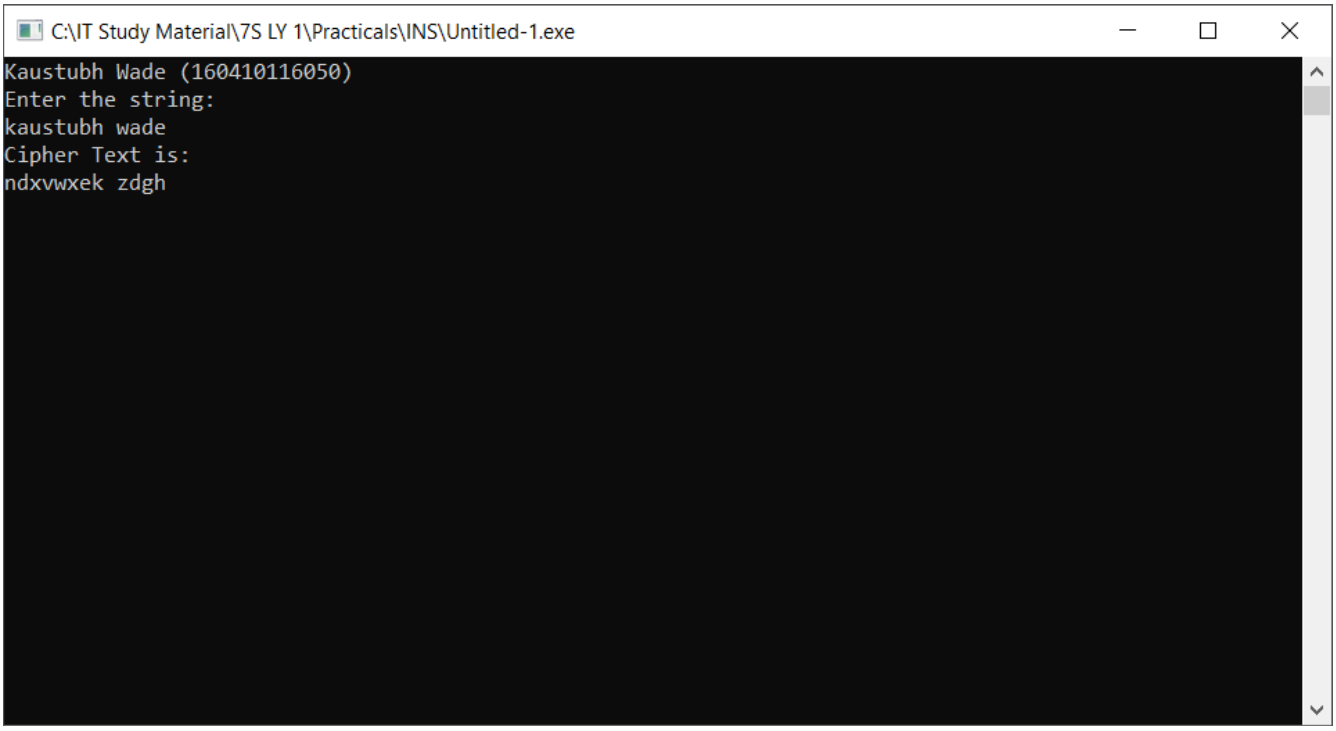
Practical 1

Implement Caesar Cipher Encryption & Decryption.

Program to Demonstrate Caesar Cipher deciphering

```c
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char plain[20];
    char cipher[20];
    int a[25];
    int b[25];
    int i, k;
    printf("Kaustubh Wade (160410116050)");
    printf("\nEnter the string:\n");
    gets(plain);
    for (i = 0; i < strlen(plain); i++)
    {
        if (plain[i] >= 65 && plain[i] <= 90)
        {
            a[i] = plain[i] - 65;
            b[i] = ((a[i] + 3) % 26);
            cipher[i] = b[i] + 65;
        }
        else if (plain[i] == ' ')
        {
            cipher[i] = plain[i];
        }
        else if (plain[i] >= 97 && plain[i] <= 122)
        {
            a[i] = plain[i] - 97;
            b[i] = ((a[i] + 3) % 26);
            cipher[i] = b[i] + 97;
        }
        else if (plain[i] >= 48 && plain[i] <= 57)
        {
            a[i] = plain[i] - 48;
            b[i] = ((a[i] + 3) % 10);
            cipher[i] = b[i] + 48;
        }
        else
            printf("Not a valid string!");
    }
    cipher[i] = '\0';
    printf("Cipher Text is:\n%s", cipher);
    scanf("%d", &i);
}
```

Output:

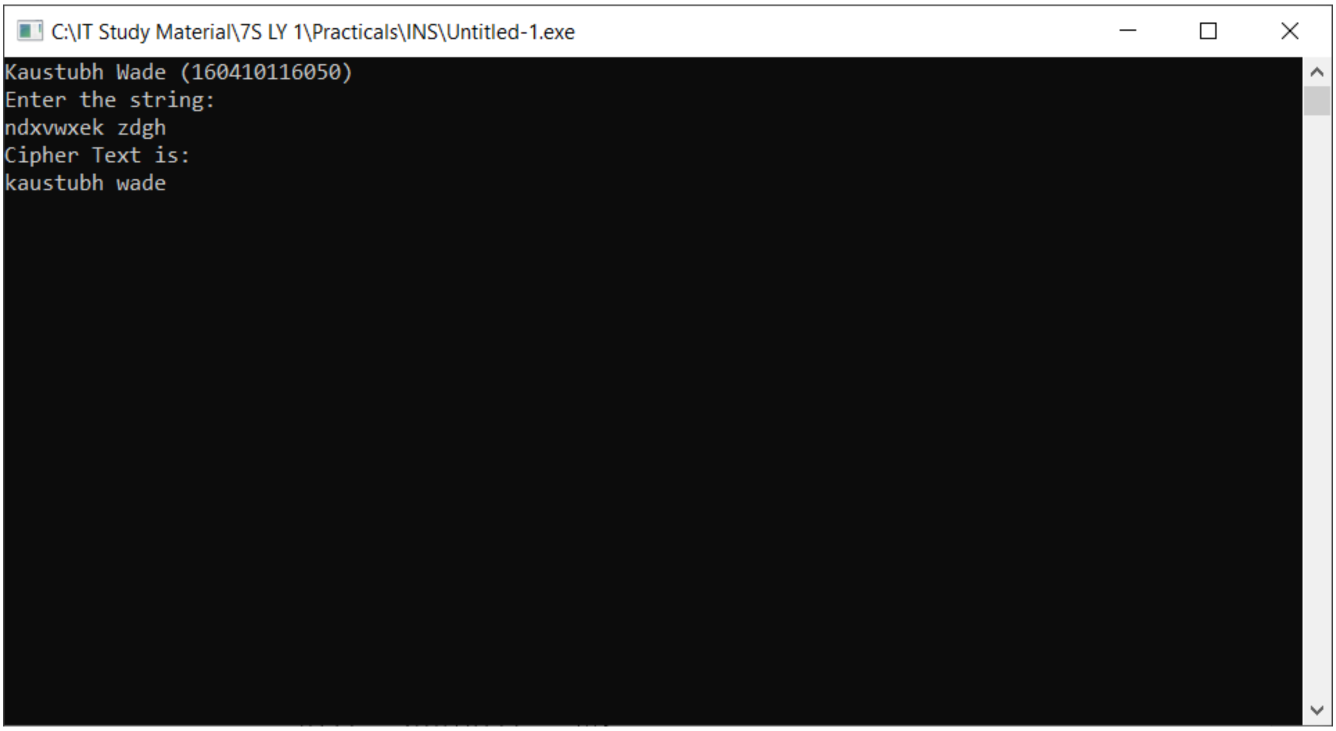


```
Kaustubh Wade (160410116050)
Enter the string:
kaustubh wade
Cipher Text is:
ndxvwxek zdgh
```

Program to Demonstrate Ceasar Cipher deciphering

```
#include <stdio.h>
#include <string.h>
void main()
{
    char plain[20];
    char cipher[20];
    int a[25];
    int b[25];
    int i, k;
    printf("Kaustubh Wade (160410116050)");
    printf("\nEnter the string:\n");
    gets(plain);
    for (i = 0; i < strlen(plain); i++)
    {
        if (plain[i] >= 65 && plain[i] <= 90)
        {
            a[i] = plain[i] - 65;
            b[i] = ((a[i] - 3 + 26) % 26);
            cipher[i] = b[i] + 65;
        }
        else if (plain[i] == ' ')
        {
            cipher[i] = plain[i];
        }
        else if (plain[i] >= 97 && plain[i] <= 122)
        {
            a[i] = plain[i] - 97;
            b[i] = ((a[i] - 3 + 26) % 26);
            cipher[i] = b[i] + 97;
        }
        else if (plain[i] >= 48 && plain[i] <= 57)
        {
            a[i] = plain[i] - 48;
            b[i] = ((a[i] - 3 + 10) % 10);
            cipher[i] = b[i] + 48;
        }
        else

            printf("Not a valid string!");
    }
    cipher[i] = '\0';
    printf("Cipher Text is:\n%s", cipher);
    scanf("%d", &i);
}
```

Output:

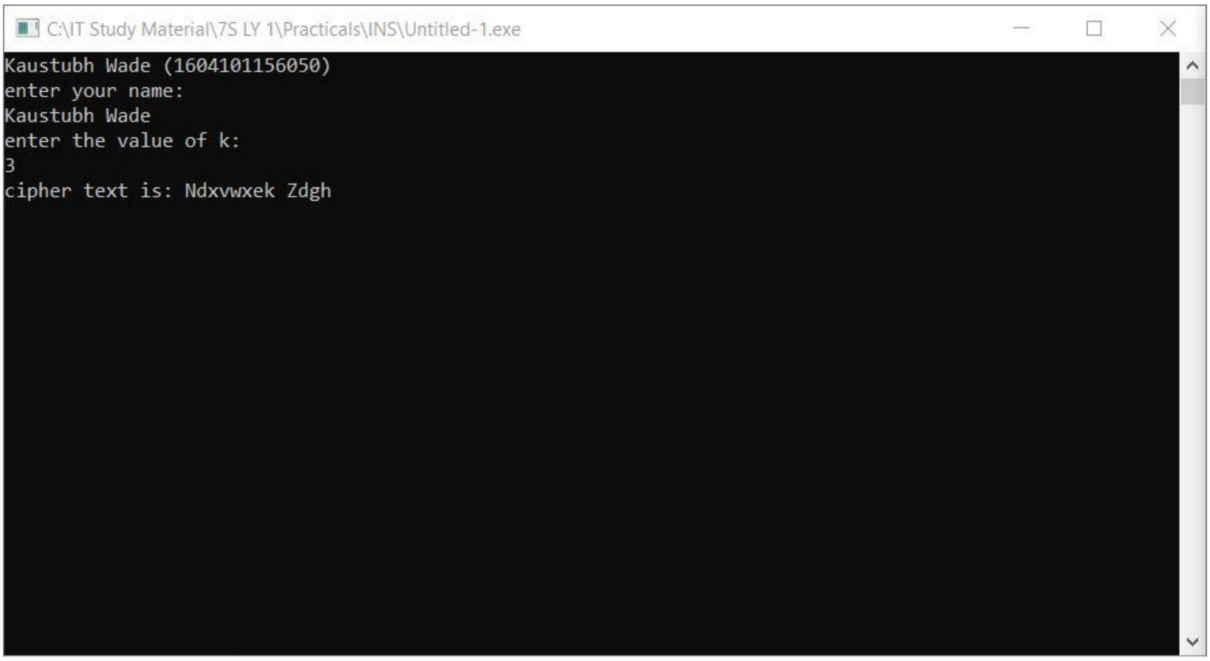


```
Kaustubh Wade (160410116050)
Enter the string:
ndxvwxek zdgh
Cipher Text is:
kaustubh wade
```

Program to Demonstrate modified version of Ceasar Cipher ciphering

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    int i, len;
    char p[20], c[20];
    int k;
    printf("Kaustubh Wade (1604101156050)\n");
    printf("enter your name:\n");
    gets(c);
    printf("enter the value of k:\n");
    scanf("%d", &k);
    len = strlen(c);
    for (i = 0; i < len; i++)
    {
        if (c[i] >= 65 && c[i] <= 90)
        {
            c[i] = c[i] - 65;
            p[i] = (c[i] + k) % 26;
            p[i] = p[i] + 65;
        }
        if (c[i] >= 97 && c[i] <= 123)
        {
            c[i] = c[i] - 97;
            p[i] = (c[i] + k) % 26;
            p[i] = p[i] + 97;
        }
        if (c[i] == ' ')
            (p[i] = ' ');
    }
    p[i] = '\0';
    printf("cipher text is: %s", p);
    getch();
}
```

Output:

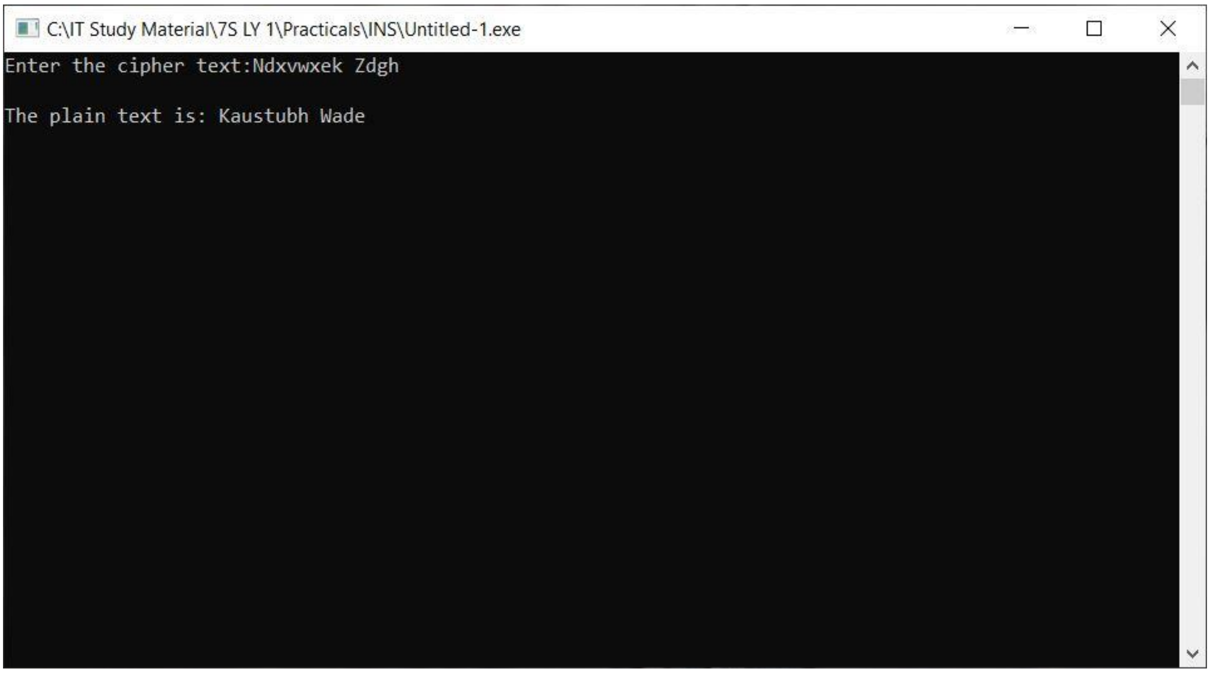


```
Kaustubh Wade (1604101156050)
enter your name:
Kaustubh Wade
enter the value of k:
3
cipher text is: Ndxvwxek Zdgh
```

Program to Demonstrate modified version of Ceasar Cipher ciphering

```
#include<stdio.h>
#include<string.h>
void main ()
{
    char a [20], b [20];
    int i;
    printf ("Enter the cipher text:");
    gets(a);
    for (i=0; i<strlen(a); i++)
    {
        if(a[i]>=65 && a[i]<=90)
            b[i]=a[i]-3;
        else if(a[i]==' ')
            b[i]=a[i];
        else
            b[i]=a[i]-3;
    }
    b[i]='\0';
    printf ("\nThe plain text is: %s", b);
    scanf("%d", &i);
}
```

Output:



```
Enter the cipher text:Ndxvwxek Zdgh

The plain text is: Kaustubh Wade
```

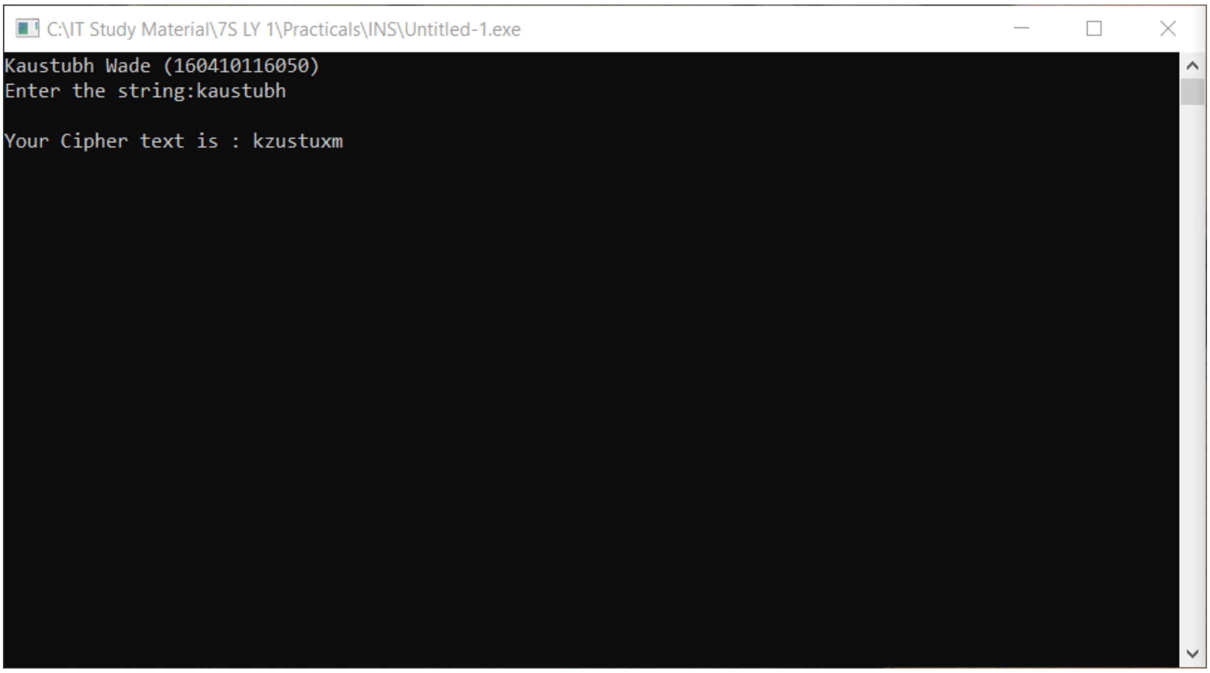
Practical 2

Implement Mono-alphabetic Cipher Encryption &Decryption.

Encryption:

```
#include <stdio.h>
#include <string.h>
void main()
{
    char pt[50], ct[50];
    int i, j, len;
    char a[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'};
    char b[] = {'z', 'x', 'c', 'd', 'e', 'f', 'i', 'm', 'o', 'j', 'k', 'l', 'g', 'n',
'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'b', 'y', 'a'};
    printf("Kaustubh Wade (160410116050)");
    printf("\nEnter the string:");
    gets(pt);
    len = strlen(pt);
    for (i = 0; i < len; i++)
    {
        for (j = 0; j < 26; j++)
        {
            if (pt[i] == a[j])
                ct[i] = b[j];
        }
    }
    ct[i] = '\0';
    printf("\nYour Cipher text is : %s", ct);
    scanf("%d",&i);
}
```

Output:

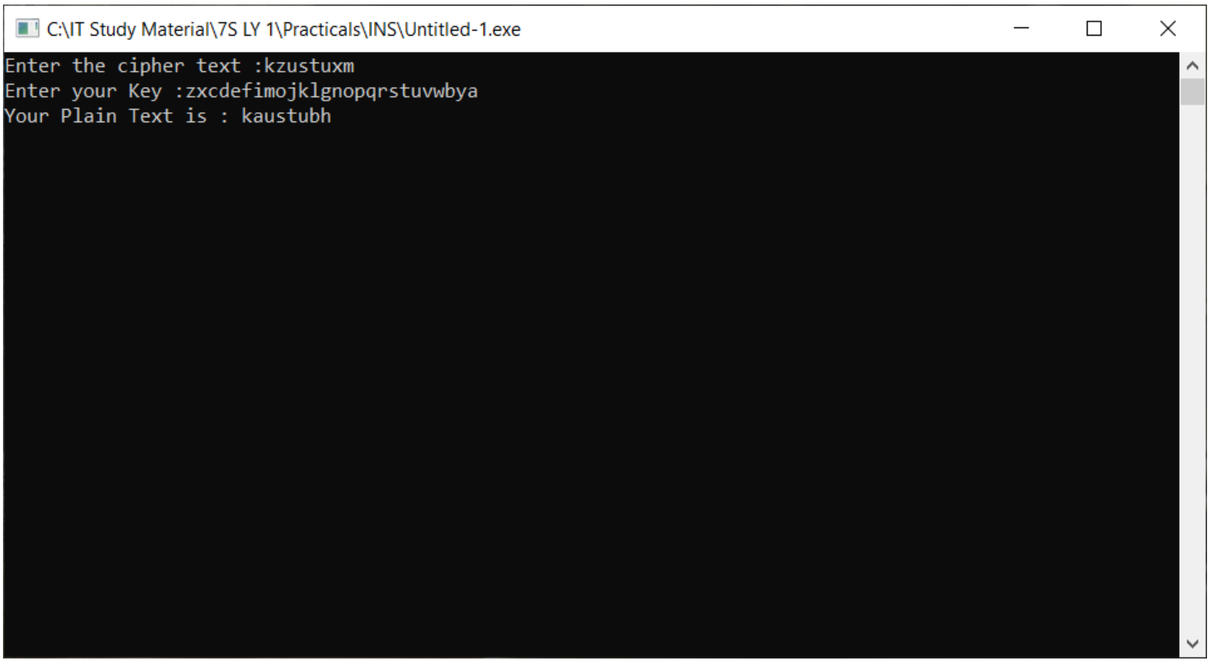


```
Kaustubh Wade (160410116050)
Enter the string:kaustubh

Your Cipher text is : kzustuxm
```

Decryption

```
#include <stdio.h>
#include<string.h>

void main()
{
    char x[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'};
    char y[28];
    char p[30], c[30];
    int i, j, len;

    printf("Enter the cipher text :");
    scanf("%[^\n]", c);
    fflush(stdin);
    printf("Enter your Key :");
    scanf("%s", y);
    fflush(stdin);
    len = strlen(c);
    for (i = 0; i<len; i++)
    {
        for (j = 0; j <= 26; j++)
        {
            if (c[i] == y[j])
            {
                p[i] = x[j];
            }
        }
    }
    p[i] = '\0';
    printf("Your Plain Text is : %s", p);
    scanf("%s", x);
}
```

Output:



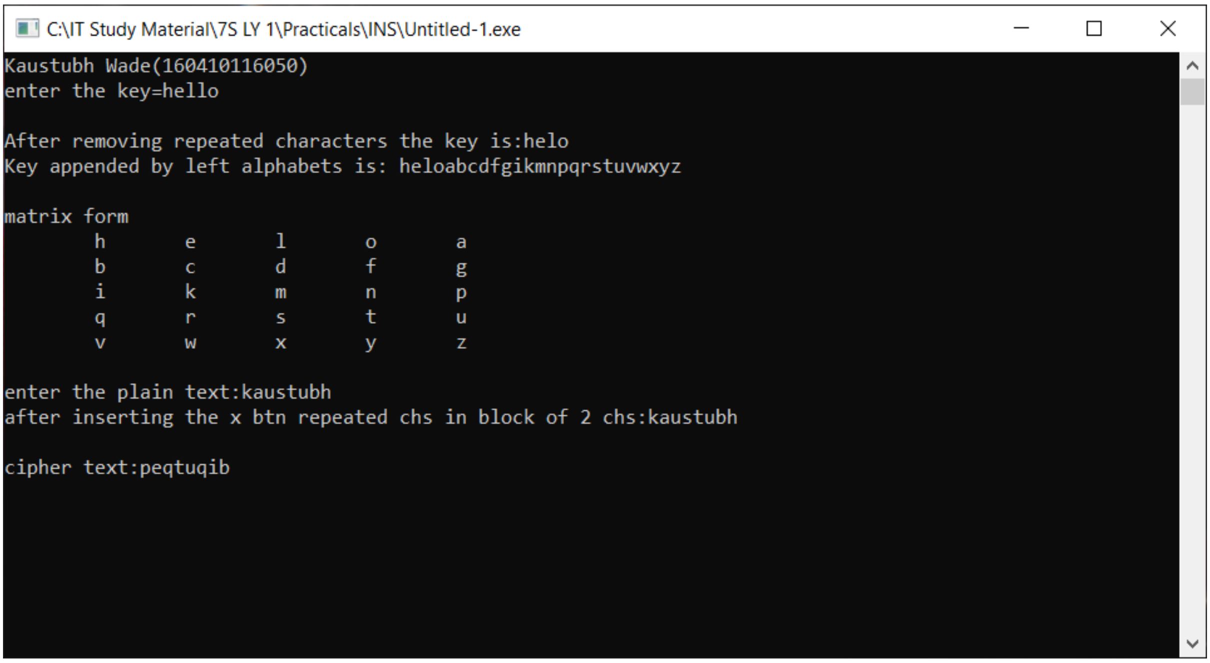
```
Enter the cipher text :kzustuxm
Enter your Key :zxcdefimojklgnopqrstuvwbya
Your Plain Text is : kaustubh
```

Practical 3

Implement Play fair Cipher Encryption-Decryption.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    char i[25], m[25], b[5][5];
    char a;
    int j, k, y, z, l, f;
    int flagi = 0, flagj = 0;
    int r;
    printf("Kaustubh Wade(160410116050)");
    printf("\nenter the key=");
    scanf("%s", i);
    l = strlen(i);
    for (j = 0; j < l; j++)
    {
        for (k = j + 1; k < l; k++)
        {
            if (i[j] == i[k])
            {
                for (r = k; r < l; r++)
                {
                    i[r] = i[r + 1];
                }
                k = j;
                l--;
            }
        }
    }

    i[l] = '\0';
    printf("\nAfter removing repeated characters the key is:%s", i);
    int u, v;
    for (k = 0; i[k] != '\0'; k++)
    {
        if (i[k] == 'i')
        {
            flagi = 1;
            u = k;
        }
        if (i[k] == 'j')
        {
            flagj = 1;
            v = k;
        }
    }
    if (flagi == 1 && flagj == 1)
    {
        if (u < v)
        {
            for (r = v; r < l; r++)
            {
                i[r] = i[r + 1];
            }
            l--;
        }
        else
        {
            for (r = u; r < l; r++)
            {
                i[r] = i[r + 1];
            }
            l--;
        }
    }
    i[l] = '\0';
    for (a = 'a'; a <= 'z'; a++)
    {
        int fg = 0;
        if (a == 'i' && flagi == 1)
        {
            a = a + 2;
        }
        if (flagj == 1 && a == 'i')
        {
            a = a + 2;
        }
        if (flagi == 0 && flagj == 0 && a == 'j')
        {
            a = a + 1;
        }
        for (k = 0; k < l; k++)
        {
            if (i[k] == a)
            {
                fg = 1;
            }
        }
        if (fg == 0)
        {
            i[l++] = a;
        }
    }
    i[l] = '\0';
    printf("\nKey appended by left alphabets is: %s", i);
    printf("\n\nmatrix form\n");
    {
        int ch = 0;
        for (y = 0; y < 5; y++)
            for (z = 0; z < 5; z++)
            {
                b[y][z] = i[ch++];
            }
    }
    for (y = 0; y < 5; y++)
    {
        for (z = 0; z < 5; z++)
        {
            printf("\t%c", b[y][z]);
        }
        printf("\n");
    }
    char pl[20], ci[20];
    printf("\nenter the plain text:");
    scanf("%s", pl);
    int plen = strlen(pl);
    for (r = 0; r < plen; r = r + 2)
    {
        if (pl[r] == pl[r + 1])
        {
            for (int s = plen; s > r; s--)
            {
                pl[s] = pl[s - 1];
            }
            pl[r + 1] = 'x';
            plen++;
        }
    }
    pl[plen] = '\0';
    int e = strlen(pl);
    if ((e % 2) != 0)
    {
        pl[e++] = 'x';
    }
    pl[e] = '\0';
    printf("after inserting the x btn repeated chs in block of 2 chs:%s", pl);
    int q = 0, t = 0;
    int row1, col1, row2, col2;
    int len = strlen(pl);
    for (q = 0; q < len; q = q + 2)
    {
        t = q;
        for (y = 0; y < 5; y++)
        {
            for (z = 0; z < 5; z++)
            {
                if (b[y][z] == pl[q])
                {
                    row1 = y;
                    col1 = z;
                }
                if (b[y][z] == pl[q + 1])
                {
                    row2 = y;
                    col2 = z;
                }
            }
        }
        if (row1 == row2)
        {
            if (col1 == 4)
            {
                col1 = 0;
                ci[t] = b[row1][col1];
                ci[t + 1] = b[row1][col2 + 1];
            }
            else if (col2 == 4)
            {
                col2 = 0;
                ci[t] = b[row1][col1 + 1];
                ci[t + 1] = b[row1][col2];
            }
            else
            {
                ci[t] = b[row1][col1 + 1];
                ci[t + 1] = b[row1][col2 + 1];
            }
        }
        else if (col1 == col2)
        {
            if (row1 == 4)
            {
                row1 = 0;
                ci[t] = b[row1][col1];
                ci[t + 1] = b[row2 + 1][col2];
            }
            else if (row2 == 4)
            {
                row2 = 0;
                ci[t] = b[row1 + 1][col1];
                ci[t + 1] = b[row2][col2];
            }
            else
            {
                ci[t] = b[row1 + 1][col1];
                ci[t + 1] = b[row2 + 1][col2];
            }
        }
        else
        {
            ci[t] = b[row1][col2];
            ci[t + 1] = b[row2][col1];
        }
    }
    printf("\n\ncipher text:");
    ci[len] = '\0';
    for (y = 0; y < strlen(ci); y++)
    {
        printf("%c", ci[y]);
    }
    getch();
}
```

Output:



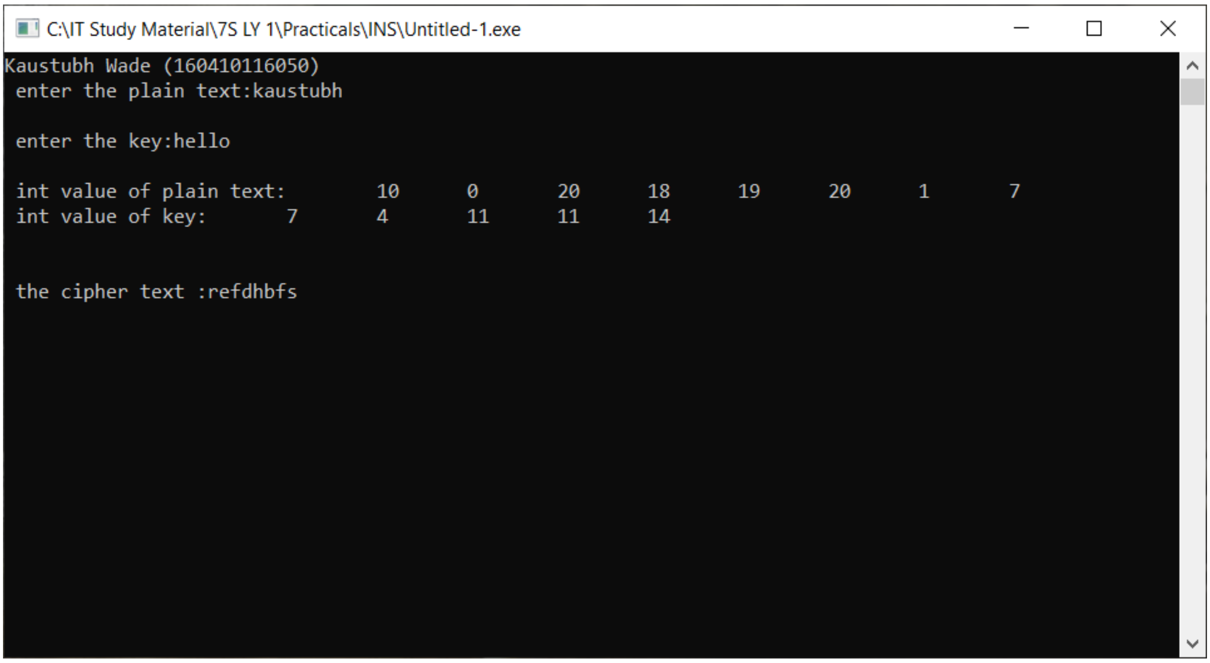
```
Kaustubh Wade(160410116050)
enter the key=hello

After removing repeated characters the key is:helo
Key appended by left alphabets is: heloabcdfgikmnpqrstuvwxyz

matrix form
        h       e       l       o       a
        b       c       d       f       g
        i       k       m       n       p
        q       r       s       t       u
        v       w       x       y       z

enter the plain text:kaustubh
after inserting the x btn repeated chs in block of 2 chs:kaustubh

cipher text:peqtuqib
```

Practical 4

Implement Polyalphabetic cipher encryption decryption

```c
#include <stdio.h>
#include <string.h>
void main()
{
    char p[50], c[50], k[50];
    int pl, ci, key;
    int i, a[50], b[50], d[50];
    printf("Kaustubh Wade (160410116050)");
    printf("\n enter the plain text:");
    scanf("%s", p);
    printf("\n enter the key:");
    scanf("%s", k);
    pl = strlen(p);
    key = strlen(k);
    for (i = 0; i < pl; i++)
    {
        a[i] = p[i] - 97;
    }
    printf("\n int value of plain text:");
    for (i = 0; i < pl; i++)
    {
        printf("\t %d", a[i]);
    }
    for (i = 0; i < key; i++)
    {
        d[i] = k[i] - 97;
    }
    printf("\n int value of key: ");
    for (i = 0; i < key; i++)
    {
        printf("\t %d", d[i]);
    }
    for (i = 0; i < pl; i++)
    {
        b[i] = (a[i] + d[i % key]) % 26;
    }
    for (i = 0; i < pl; i++)
    {
        c[i] = b[i] + 97;
    }
    int cl = strlen(p);
    c[cl] = '\0';
    printf("\n\n\n the cipher text :%s", c);
    scanf("%s", p);
}
```

Output:



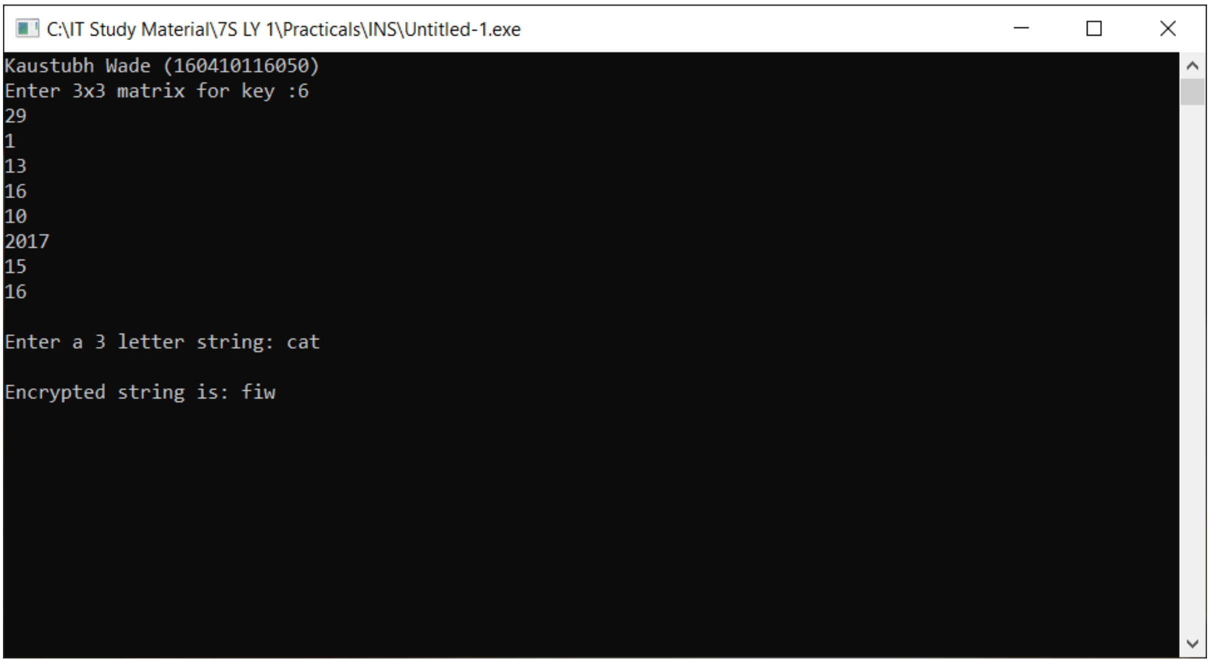
```
Kaustubh Wade (160410116050)
 enter the plain text:kaustubh

 enter the key:hello

 int value of plain text:        10      0       20      18      19      20      1       7
 int value of key:       7       4       11      11      14


 the cipher text :refdhbfs
```

Practical 5

Implement Hill cipher Encryption-Decryption

```
#include <stdio.h>
#include<math.h>
float encrypt[3][1], decrypt[3][1], a[3][3], b[3][3], mes[3][1], c[3][3];
void main()
{
    int i, j, k;
    char msg[3];
    printf("Kaustubh Wade (160410116050)");
    printf("\nEnter 3x3 matrix for key :");
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
        {
            scanf("%f", &a[i][j]);
            c[i][j] = a[i][j];
        }
    printf("\nEnter a 3 letter string: ");
    scanf("%s", msg);
    for (i = 0; i < 3; i++)
        mes[i][0] = msg[i] - 97;
    for (i = 0; i < 3; i++)
        for (j = 0; j < 1; j++)
            for (k = 0; k < 3; k++)
                encrypt[i][j] = encrypt[i][j] + a[i][k] * mes[k][j];
    printf("\nEncrypted string is: ");
    for (i = 0; i < 3; i++)
        printf("%c", (char)(fmod(encrypt[i][0], 26) + 97));
    scanf("%d", &i);
}
```

Output:



```
Kaustubh Wade (160410116050)
Enter 3x3 matrix for key :6
29
1
13
16
10
2017
15
16

Enter a 3 letter string: cat

Encrypted string is: fiw
```

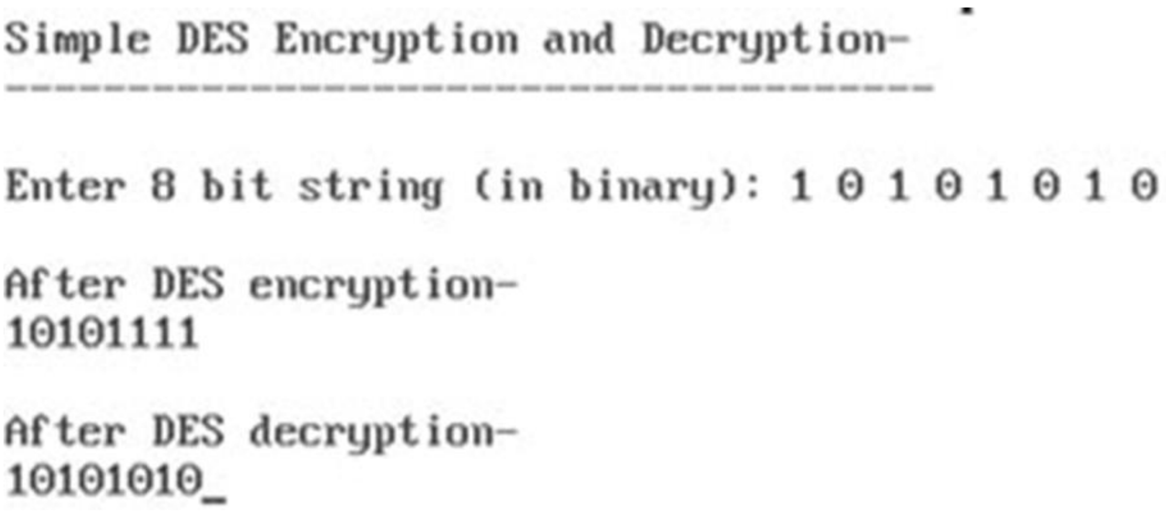
Practical 6

To Implement Simple DES or AES

```
#include <stdio.h>
int xor (int x, int y) {
    if (x == 0 && y == 1)
        return 1;
    else if (x == 1 && y == 0)
        return 1;
    else
        return 0;
} int main()
{
    int input[8], origleft[4], origright[4], encleft[4], encright[4], decleft[4], decr
ight[4], temp[4];
    int Sbox[2][2] = {{1, 0}, {0, 1}};
    printf("Simple DES Encryption and Decryption-\n");
    printf("--------------------------------------\n\n");
    printf("Enter 8 bit string (in binary): ");
    for (i = 0; i < 8; i++)
        scanf("%d", &input[i]);
    for (i = 0; i < 4; i++)
        origleft[i] = input[i];
    for (i = 4; i < 8; i++)
        origright[i - 4] = input[i];

    //Encryption
    for (i = 0; i < 4; i++)
        encleft[i] = origright[i];
    for (i = 0; i < 4; i++)
    {
        if (origright[i] == 0)
            origright[i] = 1;
        else
            origright[i] = 0;
    }
    for (i = 0; i < 4; i++)
        encright[i] = xor(origleft[i], origright[i]);
    printf("\nAfter DES encryption- \n");
    for (i = 0; i < 4; i++)
        printf("%d", encleft[i]);
    for (i = 0; i < 4; i++)
        printf("%d", encright[i]);
    printf("\n");

    //Decryption
    for (i = 0; i < 4; i++)
    {
        if (encright[i] == 0)
            encright[i] = 1;
        else
            encright[i] = 0;
    }
    for (i = 0; i < 4; i++)
```

```
        encright[i] = xor(encleft[i], encright[i]);
    for (i = 0; i < 4; i++)
        decleft[i] = encright[i];
    for (i = 0; i < 4; i++)
        decright[i] = encleft[i];
    printf("\nAfter DES decryption- \n");
    for (i = 0; i < 4; i++)
        printf("%d", decleft[i]);
    for (i = 0; i < 4; i++)
        printf("%d", decright[i]);
    return 0;
}
```

Output:



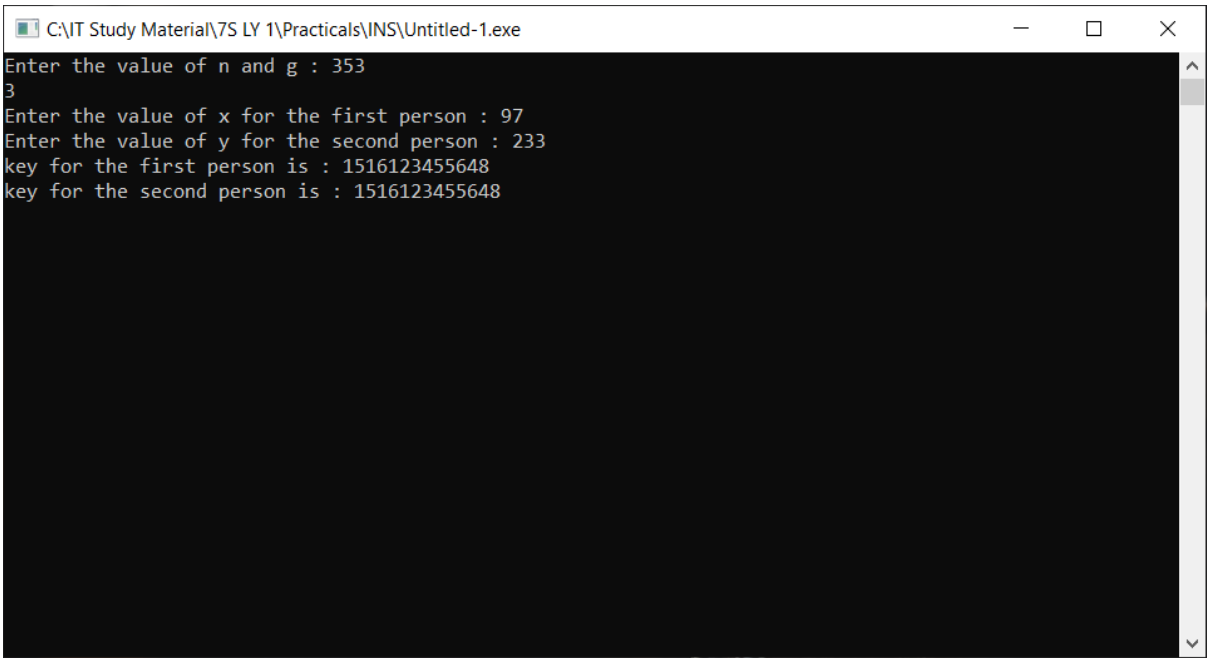
```
Simple DES Encryption and Decryption-
--------------------------------------

Enter 8 bit string (in binary): 1 0 1 0 1 0 1 0

After DES encryption-
10101111

After DES decryption-
10101010_
```

Practical 7

Implement Diffie-Hellmen Key Exchange Method

```c
#include <stdio.h>
long int power(int a, int b, int mod)
{
    long long int t;
    if (b == 1)
        return a;
    t = power(a, b / 2, mod);
    if (b % 2 == 0)
        return (t * t) % mod;
    else
        return (((t * t) % mod) * a) % mod;
}
long long int calculateKey(int a, int x, int n)
{
    return power(a, x, n);
}
void main()
{
    int n, g, x, a, y, b;
    printf("Enter the value of n and g : ");
    scanf("%d%d", &n, &g);
    printf("Enter the value of x for the first person : ");
    scanf("%d", &x);
    a = power(g, x, n);
    printf("Enter the value of y for the second person : ");
    scanf("%d", &y);
    b = power(g, y, n);
    printf("key for the first person is : %lld\n", power(b, x, n));
    printf("key for the second person is : %lld\n", power(a, y, n));
    scanf("%d",&n);
}
```

Output:



```
Enter the value of n and g : 353
3
Enter the value of x for the first person : 97
Enter the value of y for the second person : 233
key for the first person is : 1516123455648
key for the second person is : 1516123455648
```

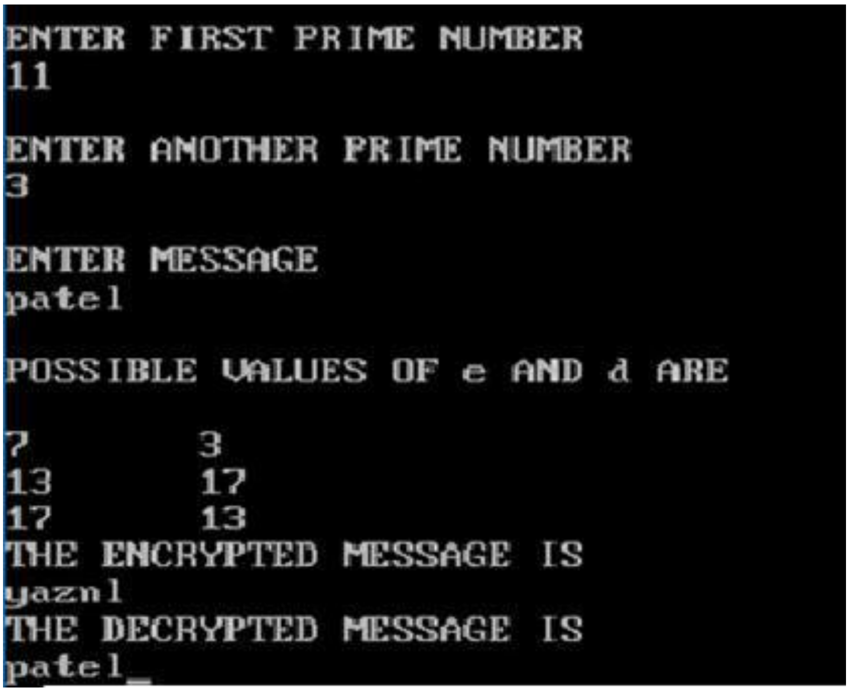
Practical 8

Implement RSA encryption-decryption algorithm

```
#include <stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
long int p, q, n, t, flag, e[100], d[100], temp[100], j, m[100], en[100], i;
char msg[100];
int prime(long int);
void ce();
long int cd(long int);
void encrypt();
void decrypt();
void main()
{
    printf("\nENTER FIRST PRIME NUMBER\n");
    scanf("%d", &p);
    flag = prime(p);
    if (flag == 0)
    {
        printf("\nWRONG INPUT\n");
        getch();
        exit(1);
    }
    printf("\nENTER ANOTHER PRIME NUMBER\n");
    scanf("%d", &q);
    flag = prime(q);
    if (flag == 0 || p == q)
    {
        printf("\nWRONG INPUT\n");
        getch();
        exit(1);
    }
    printf("\nENTER MESSAGE\n");
    fflush(stdin);
    scanf("%s", msg);
    for (i = 0; msg[i] != NULL; i++)

        TY IT - 01 Batch

                    m[i] = msg[i];
    n = p * q;
    t = (p - 1) * (q - 1);
    ce();
    printf("\nPOSSIBLE VALUES OF e AND d ARE\n");
    for (i = 0; i < j - 1; i++)
        printf("\n%ld\t%ld", e[i], d[i]);
    encrypt();
    decrypt();
    scanf("%d", i);
```

```
}
int prime(long int pr)
{
    int i;
    j = sqrt(pr);
    for (i = 2; i <= j; i++)
    {
        if (pr % i == 0)
            return 0;
    }
    return 1;
}
void ce()
{
    int k;
    k = 0;
    for (i = 2; i < t; i++)
    {
        if (t % i == 0)
            continue;
        flag = prime(i);
        if (flag == 1 && i != p && i != q)
        {
            e[k] = i;
            flag = cd(e[k]);
            if (flag > 0)
            {
                d[k] = flag;
                k++;
            }
            if (k == 99)
                break;
        }
    }

    TY IT - 01 Batch
}
long int cd(long int x)
{
    long int k = 1;
    while (1)
    {
        k = k + t;
        if (k % x == 0)
            return (k / x);
    }
}
void encrypt()
{
    long int pt, ct, key = e[0], k, len;

    i = 0;

    len = strlen(msg);
    while (i != len)
    {
        pt = m[i];
        pt = pt - 96;
        k = 1;
        for (j = 0; j < key; j++)
        {
            k = k * pt;
            k = k % n;
        }
        temp[i] = k;
        ct = k + 96;
        en[i] = ct;
        i++;
    }
    en[i] = -1;
    printf("\nTHE ENCRYPTED MESSAGE IS\n");
    for (i = 0; en[i] != -1; i++)
        printf("%c", en[i]);
}
void decrypt()
{
    long int pt, ct, key = d[0], k;
    i = 0;
    while (en[i] != -1)
    {
        ct = temp[i];
        k = 1;
        for (j = 0; j < key; j++)
        {

            TY IT - 01 Batch

                        k = k * ct;
            k = k % n;
        }
        pt = k + 96;
        m[i] = pt;
        i++;
    }
    m[i] = -1;
    printf("\nTHE DECRYPTED MESSAGE IS\n");
    for (i = 0; m[i] != -1; i++)
        printf("%c", m[i]);
}
```

Output:



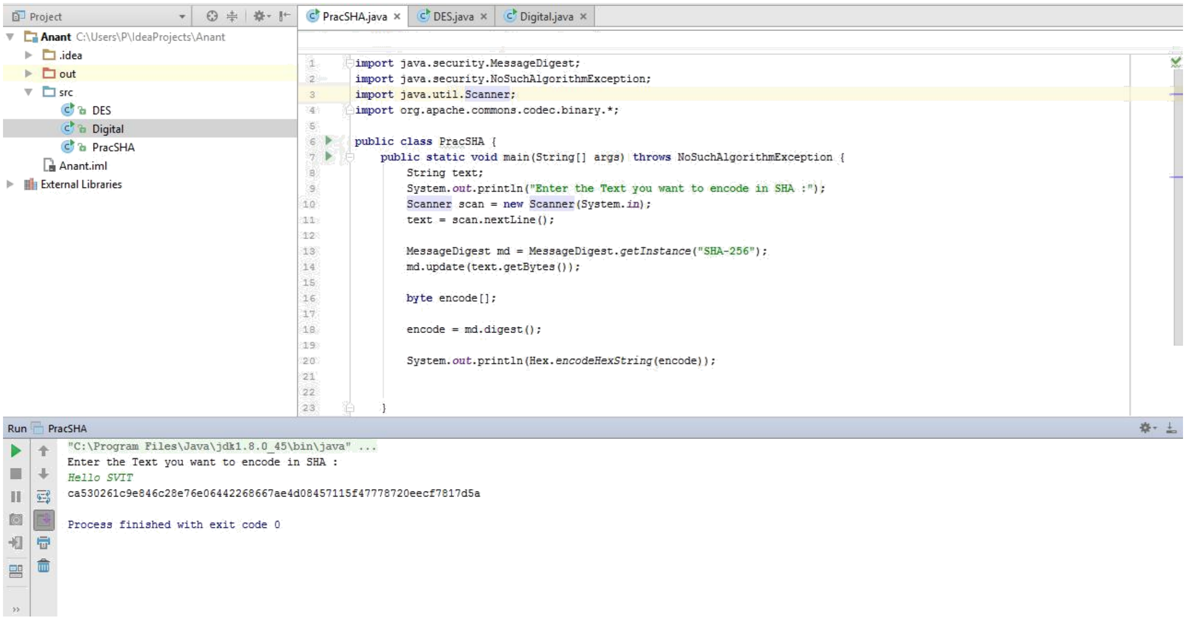
```
ENTER FIRST PRIME NUMBER
11

ENTER ANOTHER PRIME NUMBER
3

ENTER MESSAGE
patel

POSSIBLE VALUES OF e AND d ARE

7        3
13       17
17       13
THE ENCRYPTED MESSAGE IS
yaznl
THE DECRYPTED MESSAGE IS
patel_
```

Practical 9

Write a program to generate SHA-I hash

```java
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Scanner;
import org.apache.commons.codec.binary.*;
public class PracSHA
{
    public static void main(String[] args) throws NoSuchAlgorithmException
    {
        String text;
        System.out.println("Enter the Text you want to encode in SHA :");
        Scanner scan = new Scanner(System.in);
        text = scan.next();
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        md.update(text.getBytes());
        byte encode[]; encode = md.digest();
        System.out.println(Hex.encodeHexString(encode));
    }
}
```

Output:



```java
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Scanner;
import org.apache.commons.codec.binary.*;

public class PracSHA {
    public static void main(String[] args) throws NoSuchAlgorithmException {
        String text;
        System.out.println("Enter the Text you want to encode in SHA :");
        Scanner scan = new Scanner(System.in);
        text = scan.nextLine();

        MessageDigest md = MessageDigest.getInstance("SHA-256");
        md.update(text.getBytes());

        byte encode[];

        encode = md.digest();

        System.out.println(Hex.encodeHexString(encode));


    }
```

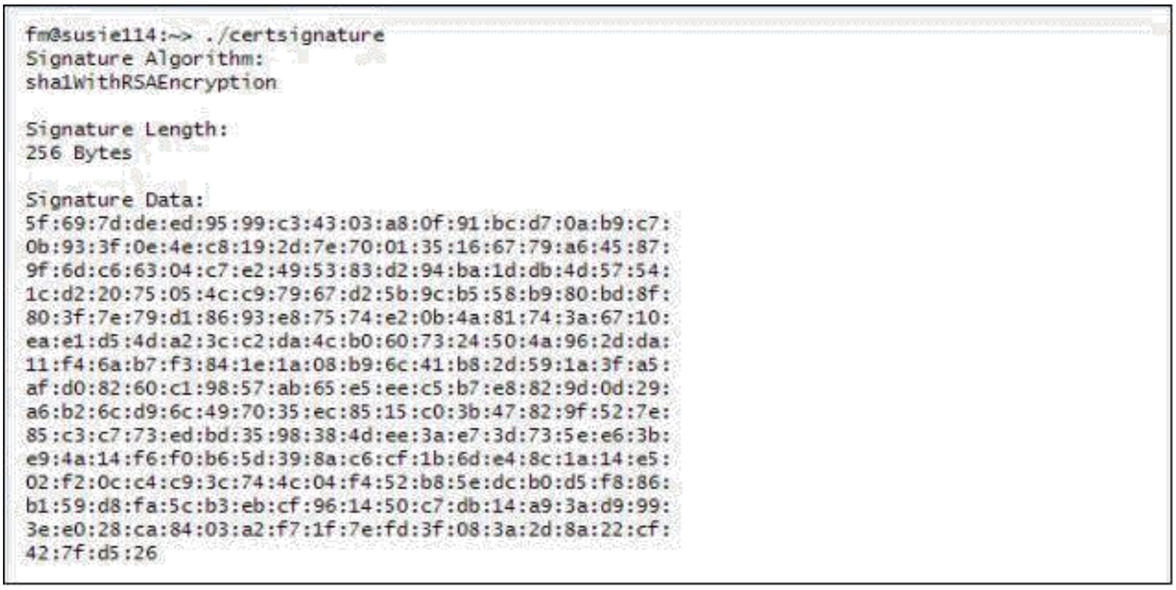
```
"C:\Program Files\Java\jdk1.8.0_45\bin\java" ...
Enter the Text you want to encode in SHA :
Hello SVIT
ca530261c9e846c28e76e06442268667ae4d08457115f47778720eecf7817d5a

Process finished with exit code 0
```

Practical 10

Implement Digital signature algorithm

```
#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/pem.h>
#include <openssl/x509.h>
int X509_signature_dump(BIO *bp, const ASN1_STRING *sig, int indent);
int main()
{
    const char cert_filestr[] = "./cert-file.pem";
    ASN1_STRING *asn1_sig = NULL;
    X509_ALGOR *sig_type = NULL;
    size_t sig_bytes = 0;
    BIO *certbio = NULL;
    BIO *outbio = NULL;
    X509 *cert = NULL;
    int ret;
    OpenSSL_add_all_algorithms();
    ERR_load_BIO_strings();
    ERR_load_crypto_strings();
    certbio = BIO_new(BIO_s_file());
    outbio = BIO_new_fp(stdout, BIO_NOCLOSE);
    ret =
        BIO_read_filename(certbio, cert_filestr);
    if (!(cert = PEM_read_bio_X509(certbio, NULL, 0, NULL)))
    {
        BIO_printf(outbio, "Error loading cert into memory\n");
        exit(-1);
    }
    sig_type = cert->sig_alg;
    asn1_sig = cert->signature;
    sig_bytes = asn1_sig->length;
    BIO_printf(outbio, "Signature Algorithm:\n");
    if (i2a_ASN1_OBJECT(outbio, sig_type->algorithm) <= 0)
        BIO_printf(outbio, "Error getting the signature algorithm.\n");
    else
        BIO_puts(outbio, "\n\n");
    BIO_printf(outbio, "Signature Length:\n%d Bytes\n\n", sig_bytes);
    BIO_printf(outbio, "Signature Data:");
    if (X509_signature_dump(outbio, asn1_sig, 0) != 1) BIO_printf(outbio, "Error printing the signature data\n");
    X509_free(cert);
    BIO_free_all(certbio);
    BIO_free_all(outbio);
    exit(0);
}
int X509_signature_dump(BIO *bp, const ASN1_STRING *sig, int indent)
{
    const unsigned char *s;
    int i, n;
    n = sig->length;
    s = sig->data;
    for (i = 0; i < n; i++)
    {
        if ((i % 18) == 0)
        {
            if (BIO_write(bp, "\n", 1) <= 0)
                return 0;
            if (BIO_indent(bp, indent, indent) <= 0)
                return 0;
        }
        if (BIO_printf(bp, "%02x%s", s[i], ((i + 1) == n) ? "" : ":") <= 0)
            return 0;
    }
    if (BIO_write(bp, "\n", 1) != 1)
        return 0;
    return 1;
}
```

Output:



```
fm@susie114:~> ./certsignature
Signature Algorithm:
sha1WithRSAEncryption

Signature Length:
256 Bytes

Signature Data:
5f:69:7d:de:ed:95:99:c3:43:03:a8:0f:91:bc:d7:0a:b9:c7:
0b:93:3f:0e:4e:c8:19:2d:7e:70:01:35:16:67:79:a6:45:87:
9f:6d:c6:63:04:c7:e2:49:53:83:d2:94:ba:1d:db:4d:57:54:
1c:d2:20:75:05:4c:c9:79:67:d2:5b:9c:b5:58:b9:80:bd:8f:
80:3f:7e:79:d1:86:93:e8:75:74:e2:0b:4a:81:74:3a:67:10:
ea:e1:d5:4d:a2:3c:c2:da:4c:b0:60:73:24:50:4a:96:2d:da:
11:f4:6a:b7:f3:84:1e:1a:08:b9:6c:41:b8:2d:59:1a:3f:a5:
af:d0:82:60:c1:98:57:ab:65:e5:ee:c5:b7:e8:82:9d:0d:29:
a6:b2:6c:d9:6c:49:70:35:ec:85:15:c0:3b:47:82:9f:52:7e:
85:c3:c7:73:ed:bd:35:98:38:4d:ee:3a:e7:3d:73:5e:e6:3b:
e9:4a:14:f6:f0:b6:5d:39:8a:c6:cf:1b:6d:e4:8c:1a:14:e5:
02:f2:0c:c4:c9:3c:74:4c:04:f4:52:b8:5e:dc:b0:d5:f8:86:
b1:59:d8:fa:5c:b3:eb:cf:96:14:50:c7:db:14:a9:3a:d9:99:
3e:e0:28:ca:84:03:a2:f7:1f:7e:fd:3f:08:3a:2d:8a:22:cf:
42:7f:d5:26
```

Practical 11

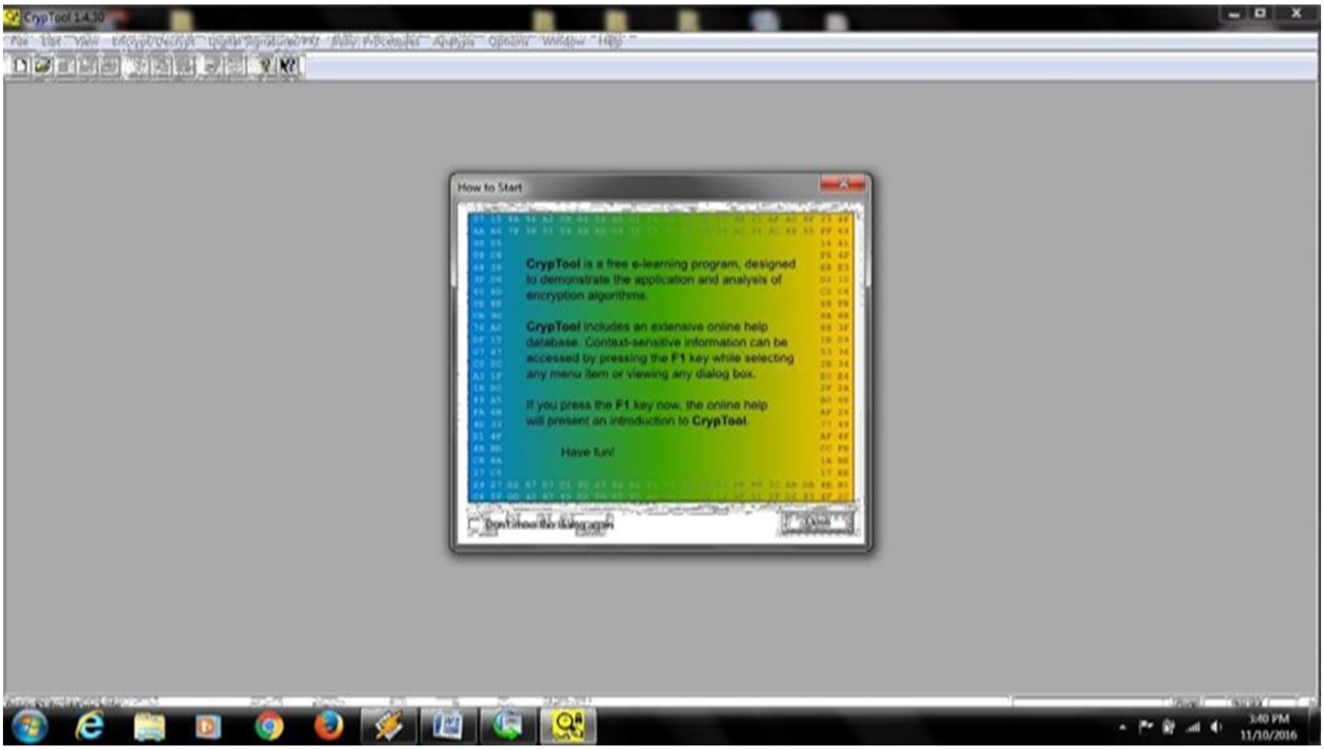
Perform various encryption-decryption techniques with cryptool.

What is CrypTool?

- CrypTool is an open source e-learning tool illustrating cryptographic and cryptanalytic concepts.
- CrypTool implements more than 300 algorithms. Users can adjust these with own parameters. The graphical interface, online documentation, analytic tools and algorithms of Crypt Tool introduce users to the field of cryptography.
- Classical ciphers are available alongside asymmetric cryptography including RSA, elliptic curve cryptography, digital signatures, homomorphism encryption, and Diffie– Hellman key exchange, many of which are visualized by animations.
-

Steps to perform encryption using CrypTool

Step 1: Start CrypTool



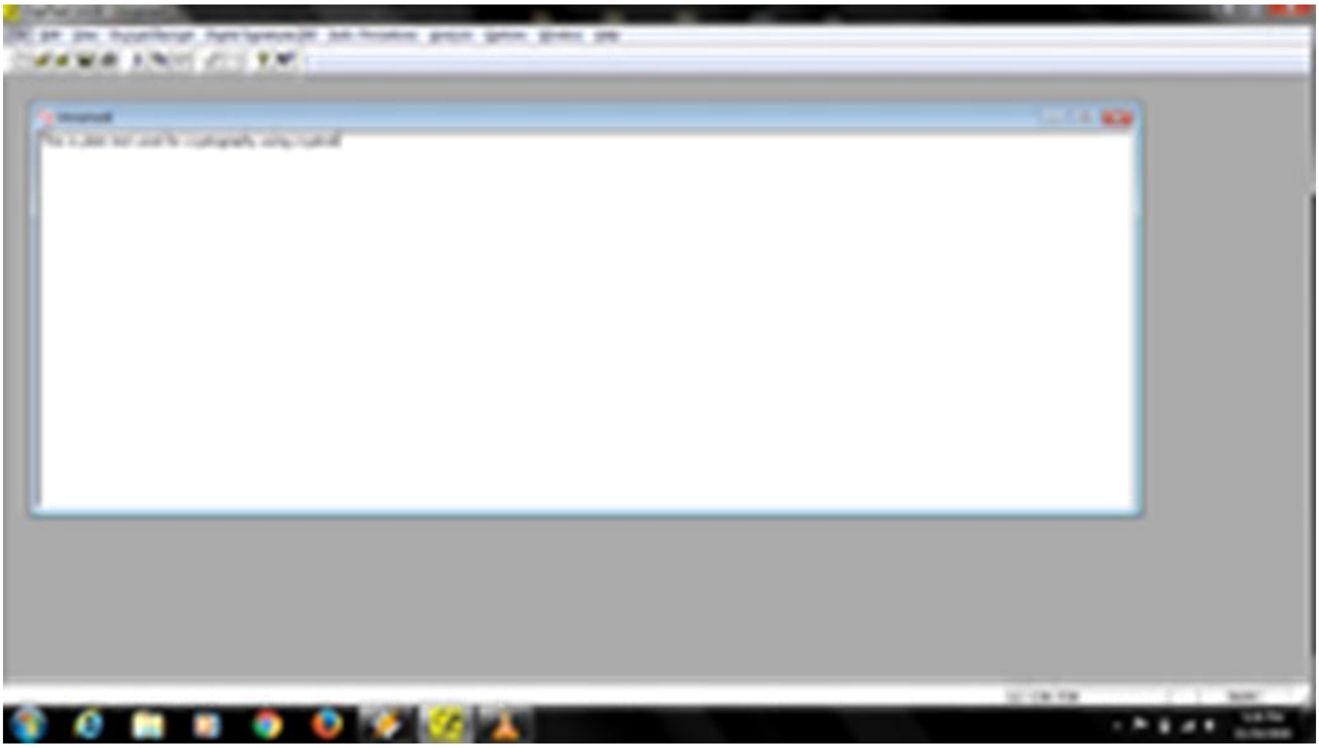
CrypTool is a free e-learning program, designed to demonstrate the application and analysis of encryption algorithms.

CrypTool includes an extensive online help database. Context-sensitive information can be accessed by pressing the **F1** key while selecting any menu item or viewing any dialog box.

If you press the **F1** key now, the online help will present an introduction to **CrypTool**.

Have fun!

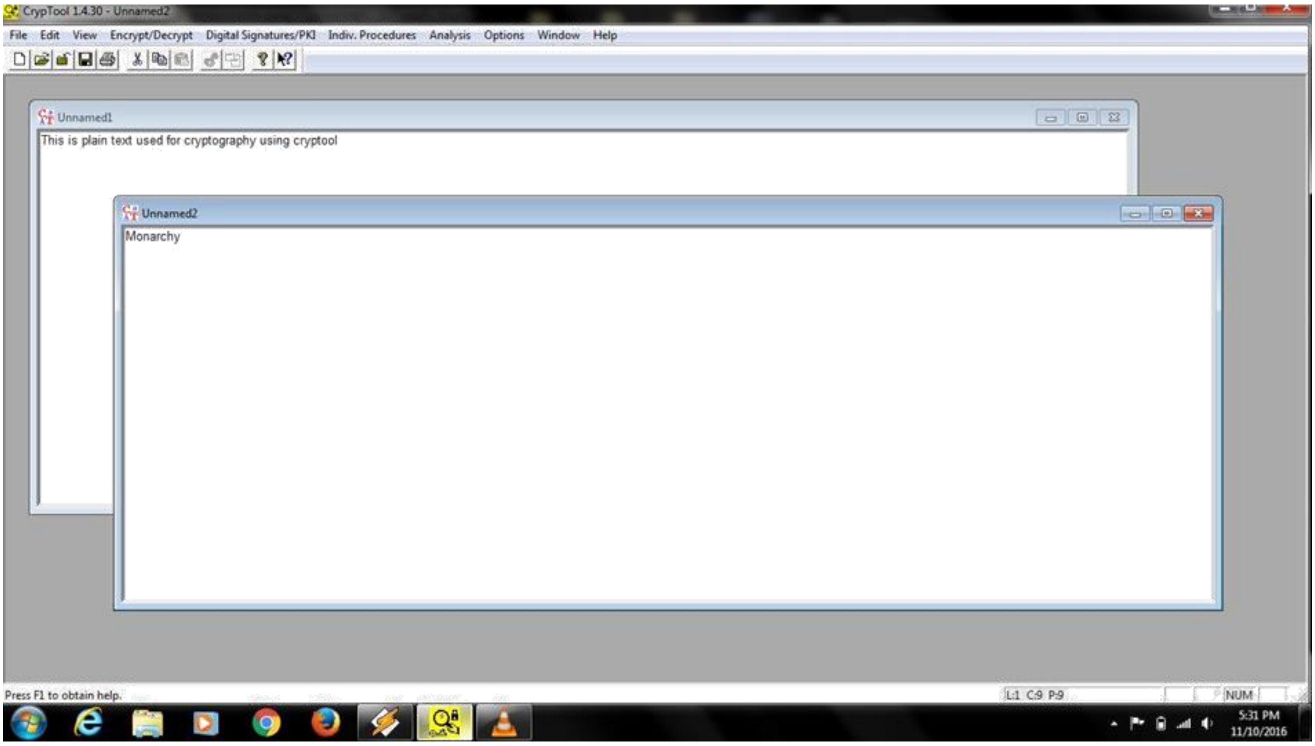
Step 2: Create the document on which you want to apply encryption algorithm.



Unnamed1

This is plain text used for cryptography using cryptool

Step 3: Create the document for the key used for encryption-decryption technique.



CrypTool 1.4.30 - Unnamed2

Unnamed1

This is plain text used for cryptography using cryptool

Unnamed2

Monarchy

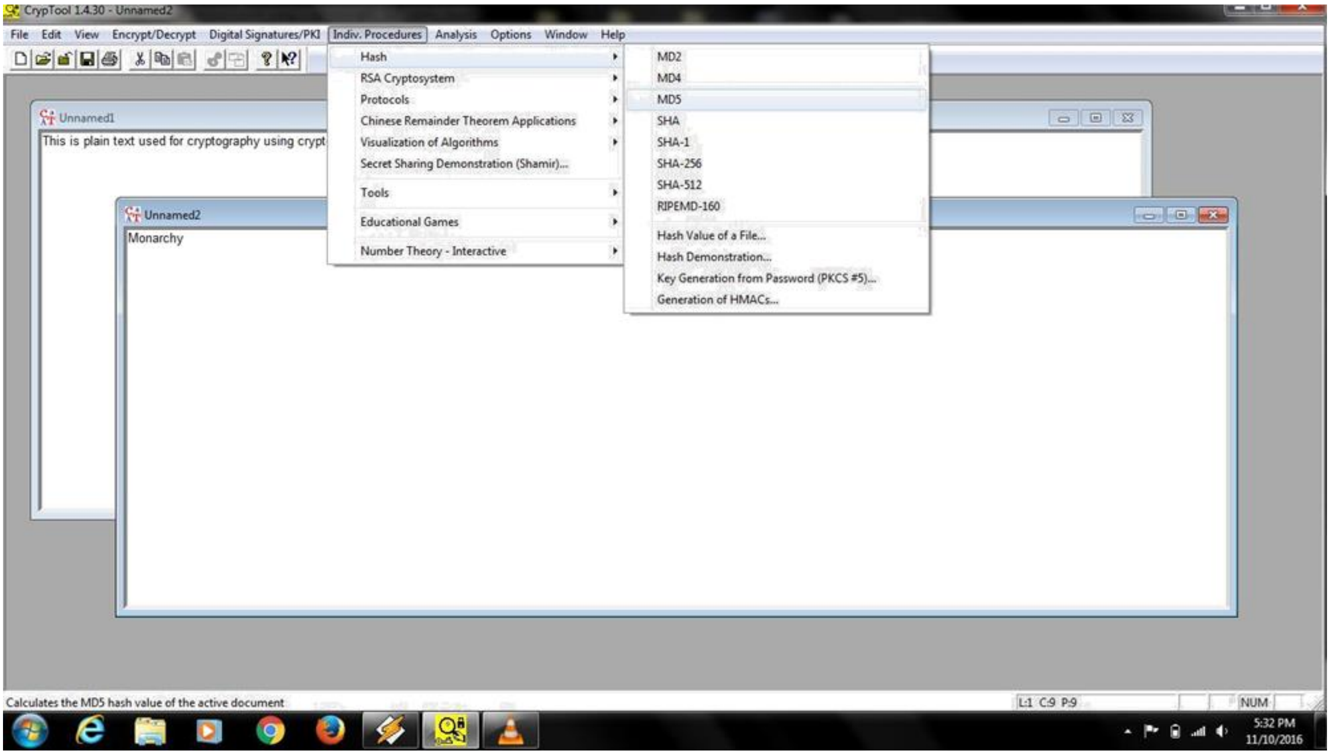
Press F1 to obtain help.

L:1 C:9 P:9

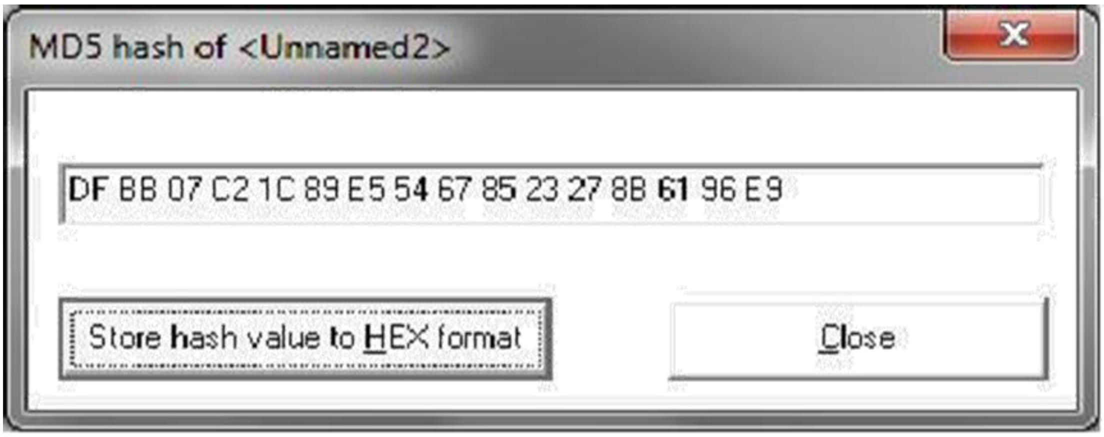
NUM

5:31 PM
11/10/2016

Step 4: Click on Indiv. Procedures and select the algorithm to generate hash value.

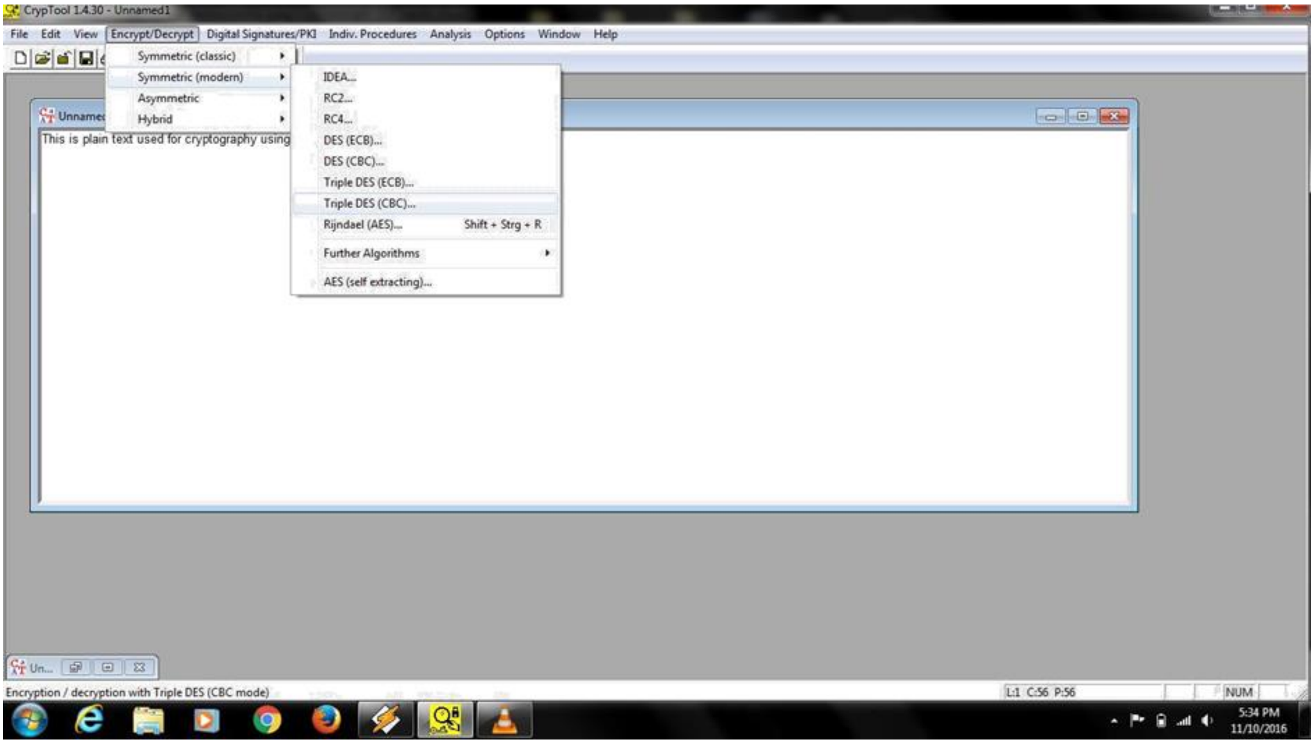

CrypTool 1.4.30 - Unnamed2
File Edit View Encrypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help
Hash
RSA Cryptosystem
Protocols
Chinese Remainder Theorem Applications
Visualization of Algorithms
Secret Sharing Demonstration (Shamir)...
Tools
Educational Games
Number Theory - Interactive
MD2
MD4
MD5
SHA
SHA-1
SHA-256
SHA-512
RIPEMD-160
Hash Value of a File...
Hash Demonstration...
Key Generation from Password (PKCS #5)...
Generation of HMACs...
Unnamed1
This is plain text used for cryptography using crypt
Unnamed2
Monarchy
Calculates the MD5 hash value of the active document
L:1 C:9 P:9
NUM
5:32 PM
11/10/2016

Step 5: Click on Store hash value to Hex format.



DF BB 07 C2 1C 89 E5 54 67 85 23 27 8B 61 96 E9

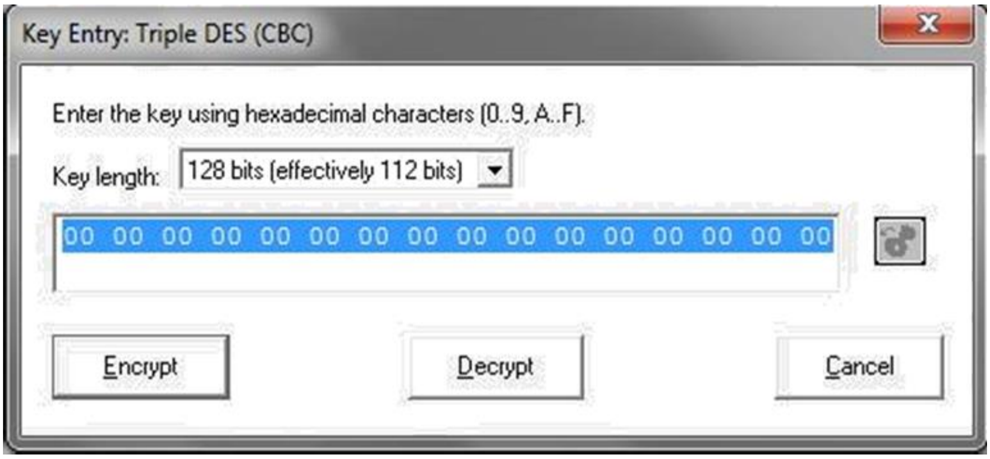
Step 6: Click on Encrypt/Decrypt and select the algorithm you want to apply for encryption.



This is plain text used for cryptography using

Encryption / decryption with Triple DES (CBC mode)

Step 7: Dialog box is appeared as shown below.

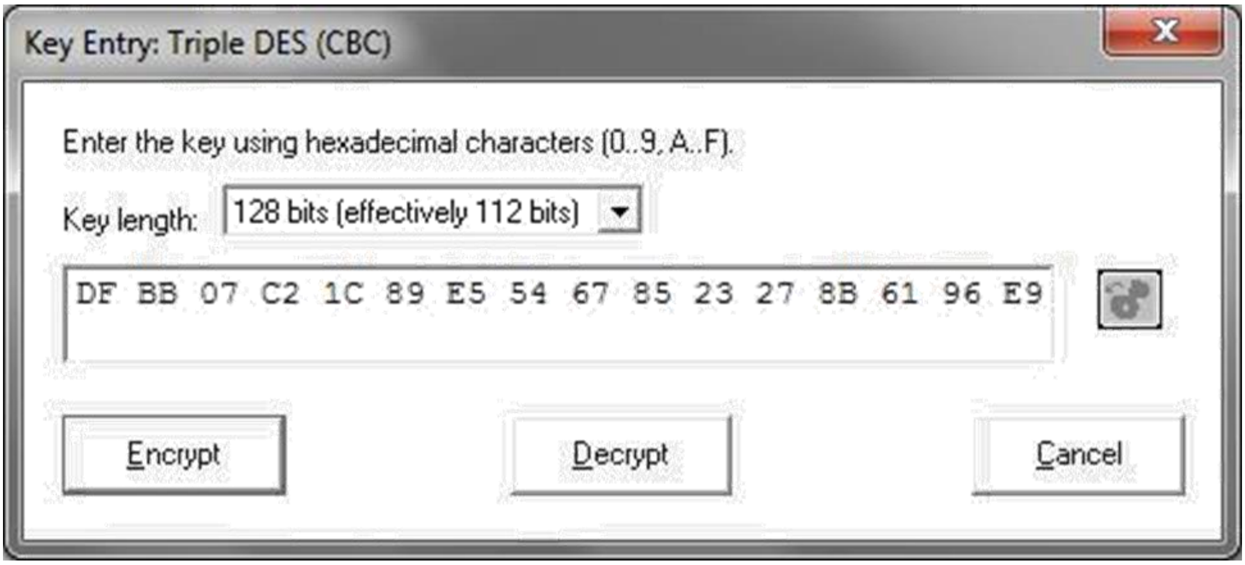

Key Entry: Triple DES (CBC)

Enter the key using hexadecimal characters (0..9, A..F).

Key length: 128 bits (effectively 112 bits)

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Step 8: Replace the key by the hash value and click on Encrypt



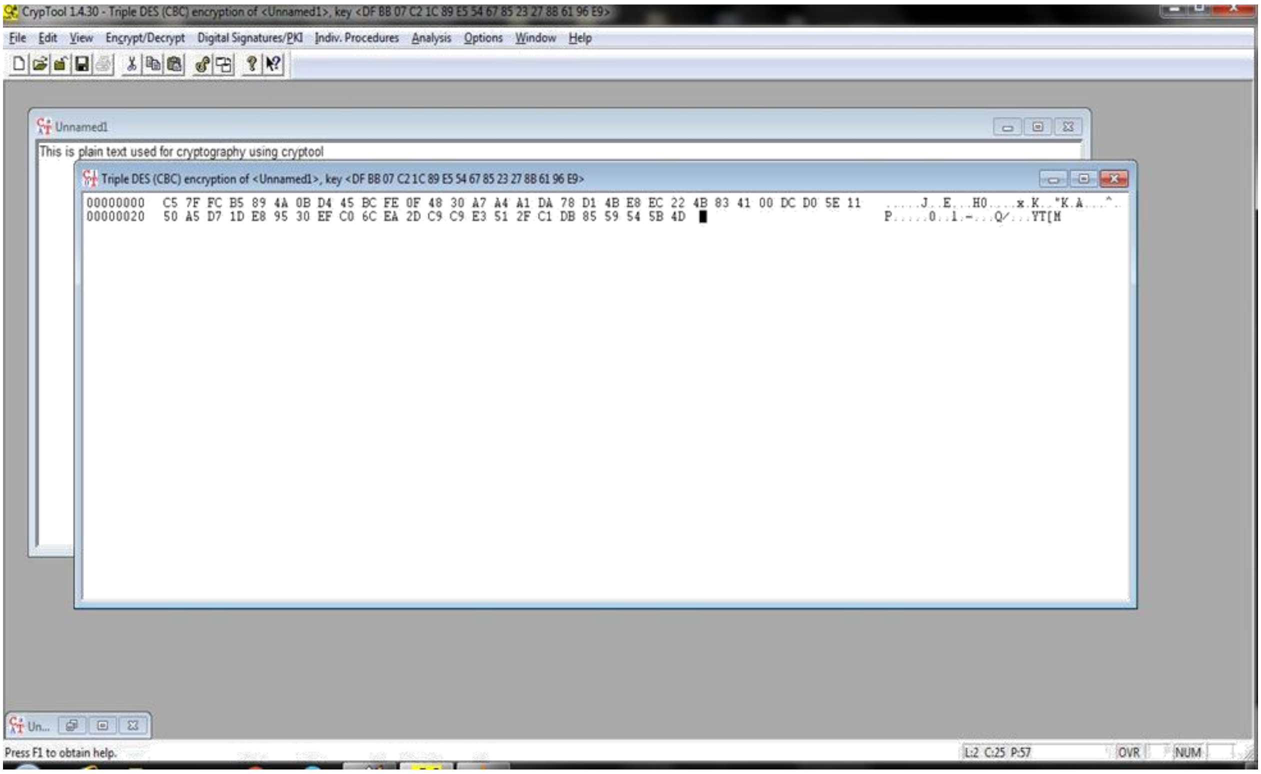
Key Entry: Triple DES (CBC)

Enter the key using hexadecimal characters (0..9, A..F).

Key length: 128 bits (effectively 112 bits)

```
DF BB 07 C2 1C 89 E5 54 67 85 23 27 8B 61 96 E9
```

Output:



CrypTool 1.4.30 - Triple DES (CBC) encryption of <Unnamed1>, key <DF BB 07 C2 1C 89 E5 54 67 85 23 27 88 61 96 E9>

Unnamed1

This is plain text used for cryptography using cryptool

Triple DES (CBC) encryption of <Unnamed1>, key <DF BB 07 C2 1C 89 E5 54 67 85 23 27 88 61 96 E9>

```
00000000   C5 7F FC B5 89 4A 0B D4 45 BC FE 0F 48 30 A7 A4 A1 DA 78 D1 4B E8 EC 22 4B 83 41 00 DC D0 5E 11    .....J..E...H0....x.K.."K.A....^.
00000020   50 A5 D7 1D E8 95 30 EF C0 6C EA 2D C9 C9 E3 51 2F C1 DB 85 59 54 5B 4D █                          P.....0..l.-...Q/...YT[M
```

Press F1 to obtain help. L:2 C:25 P:57 OVR NUM

Practical 12

Study and use Wire-Shark for various network protocols

Packer sniffer

The basic tool for observing the messages exchanged between executing protocol entities is called a packet sniffer. As the name suggests, a packet sniffer captures (“sniffs”) messages being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a copy of packets that are sent / received from/by application and protocols executing on your machine.

At the right of Figure 1 are the protocols and applications that normally run on your computer. The packet sniffer, shown within the dashed rectangle in Figure 1 is an addition to the usual software in your computer, and consists of two parts. The packet capture library receives a copy of every link-layer frame that is sent from or received by your computer. Messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. In Figure 1, the assumed physical media is an Ethernet, and so all upper layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you all messages sent/received from/by all protocols and applications executing in your computer.

The second component of a packet sniffer is the packet analyzer, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must “understand” the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by the HTTP protocol in Figure 1. The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. Finally, it understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string “GET,” “POST,” or “HEAD”.

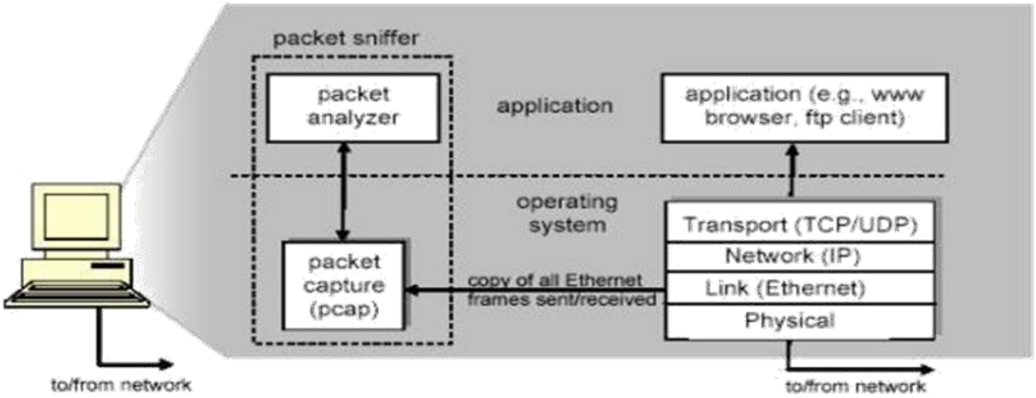


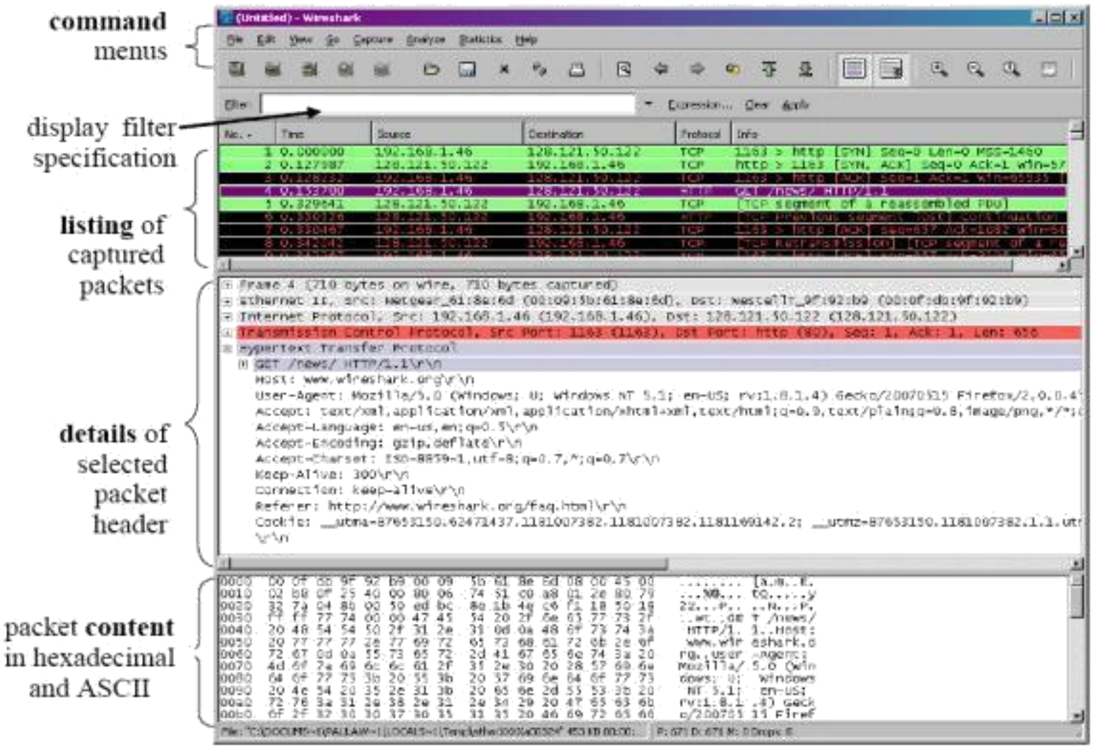
Figure 1: Packet sniffer structure

We will be using the Wireshark packet sniffer [http://www.wireshark.org/] for these labs, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. (Technically speaking, Wireshark is a packet analyzer that uses a packet capture library in your computer). Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac computers.

It's an ideal packet analyzer for our labs – it is stable, has a large user base and well-documented support that includes a user-guide (http://www.wireshark.org/docs/wsug_html_chunked/),manpages (http://www.wireshark.org/docs/man-pages/), and a detailed FAQ (http://www.wireshark.org/faq.html), rich functionality that includes the capability to analyze hundreds of protocols, and a well-designed user interface. It operates in computers using Ethernet, Token-Ring, FDDI, serial (PPP and SLIP), 802.11 wireless LANs, and ATM connections (if the OS on which it's running allows Wireshark to do so).

Running Wireshark

When you run the Wireshark program, the Wireshark graphical user interface shown in Figure 2 will de displayed. Initially, no data will be displayed in the various windows.


command menus
display filter specification
listing of captured packets
details of selected packet header
packet content in hexadecimal and ASCII
(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply

```
No. -  Time        Source            Destination       Protocol  Info
   1 0.000000    192.168.1.46      128.121.50.122    TCP       1163 > http [SYN] Seq=0 Len=0 MSS=1460
   2 0.127987    128.121.50.122    192.168.1.46      TCP       http > 1163 [SYN, ACK] Seq=0 Ack=1 Win=57
   3             192.168.1.46      128.121.50.122    TCP       1163 > http [ACK] Seq=1 Ack=1 Win=65535
   4             192.168.1.46      128.121.50.122    HTTP      GET /news/ HTTP/1.1
   5 0.329641    128.121.50.122    192.168.1.46      TCP       [TCP segment of a reassembled PDU]
   6             128.121.50.122    192.168.1.46      HTTP      [TCP Previous segment lost] Continuation
   7             192.168.1.46      128.121.50.122    TCP       1163 > http [ACK] Seq=657 Ack=1082 Win=64
   8             128.121.50.122    192.168.1.46      TCP       [TCP Retransmission] [TCP segment of a re
```

```
⊞ Frame 4 (710 bytes on wire, 710 bytes captured)
⊞ Ethernet II, Src: Netgear_61:8e:6d (00:09:5b:61:8e:6d), Dst: Westell_9f:92:b9 (00:0f:db:9f:92:b9)
⊞ Internet Protocol, Src: 192.168.1.46 (192.168.1.46), Dst: 128.121.50.122 (128.121.50.122)
⊞ Transmission Control Protocol, Src Port: 1163 (1163), Dst Port: http (80), Seq: 1, Ack: 1, Len: 656
⊟ Hypertext Transfer Protocol
  ⊞ GET /news/ HTTP/1.1\r\n
    Host: www.wireshark.org\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4
    Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;
    Accept-Language: en-us,en;q=0.5\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    Referer: http://www.wireshark.org/faq.html\r\n
    Cookie: __utma=87653150.62471437.1181007382.1181007382.1181169142.2; __utmz=87653150.1181007382.1.1.utr
    \r\n
```

```
0000  00 0f db 9f 92 b9 00 09  5b 61 8e 6d 08 00 45 00   ........ [a.m..E.
0010  02 b8 0f 25 40 00 80 06  74 51 c0 a8 01 2e 80 79   ...%@... tQ.....y
0020  32 7a 04 8b 00 50 ed bc  8e 1b 4e c6 f1 18 50 18   2z...P.. ..N...P.
0030  ff ff 77 74 00 00 47 45  54 20 2f 6e 65 77 73 2f   ..wt..GE T /news/
0040  20 48 54 54 50 2f 31 2e  31 0d 0a 48 6f 73 74 3a    HTTP/1. 1..Host:
0050  20 77 77 77 2e 77 69 72  65 73 68 61 72 6b 2e 6f    www.wir eshark.o
0060  72 67 0d 0a 55 73 65 72  2d 41 67 65 6e 74 3a 20   rg..User -Agent: 
```

Figure 2: Wireshark Graphical User Interface

The Wireshark interface has five major components:

The command menusare standard pulldown menus located at the top of the window. Of interest to us now are the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data, and exit the Wireshark application. The Capture menu allows you to begin packet capture.

The packet-listing window displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is not a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.

The packet-header details window provides details about the packet selected (highlighted) in the packet listing window. (To select a packet in the packet listing window, place the cursor over the packet's one-line summary in the packet listing window and click with the left mouse button.). These details include information about the Ethernet frame and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the right-pointing or down-pointing arrowhead to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can similarly be expanded or minimized. Finally, details about the highest level protocol that sent or received this packet are also provided.

The packet-contents window displays the entire contents of the captured frame, in both ASCII and hexadecimal format.

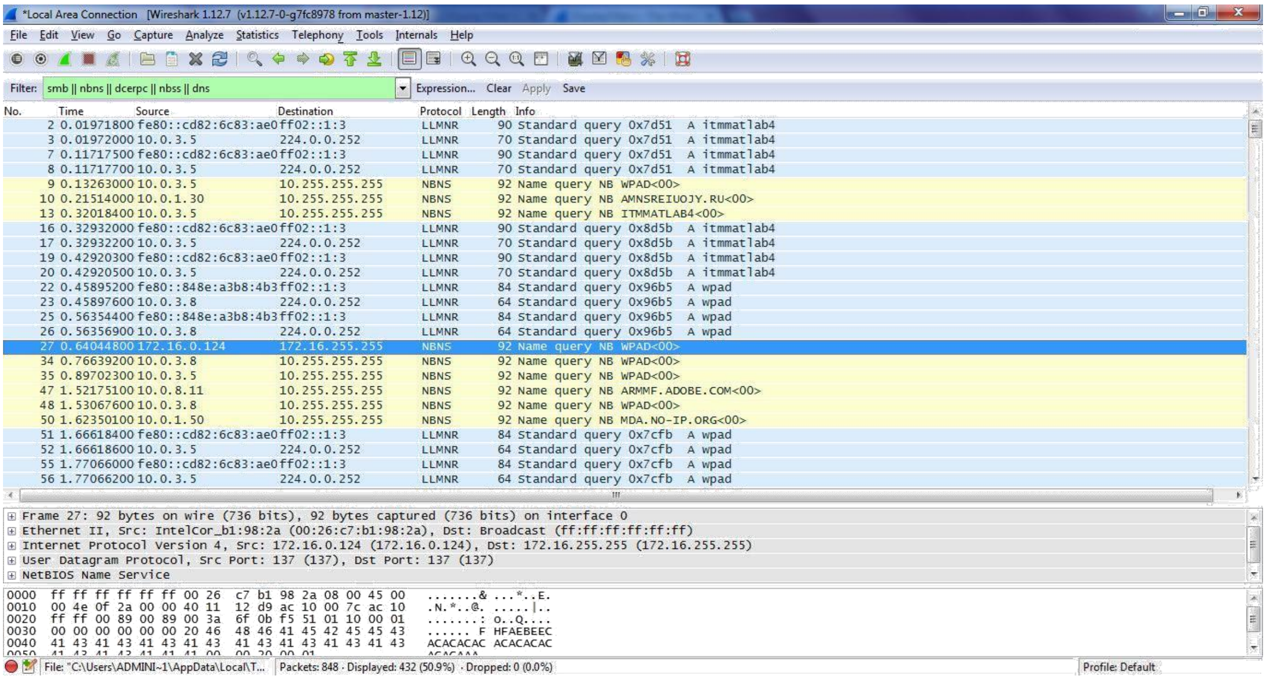
Towards the top of the Wireshark graphical user interface, is the packet display filter field, into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing window (and hence the packet-header and packet-contents windows). In the example below, we'll use the packet-display filter field to have Wireshark hide (not display) packets except those that correspond to HTTP messages.

Match Packets Containing a Particular Sequence

Wireshark displays the data contained by a packet (which is currently selected) at the bottom of the window. Sometimes, while debugging a problem, it is required to filter packets based on a particular byte sequence. We can easily do that using Wireshark.

For example, TCP packets containing the 01:01:04 byte sequence can be filtered using the following way:

tcp contains 01:01:04


*Local Area Connection [Wireshark 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save

Filter: smb || nbns || dcerpc || nbss || dns

```
No.  Time       Source                    Destination      Protocol Length Info
   2 0.01971800 fe80::cd82:6c83:ae0 ff02::1:3        LLMNR    90 Standard query 0x7d51  A itmmatlab4
   3 0.01972000 10.0.3.5            224.0.0.252      LLMNR    70 Standard query 0x7d51  A itmmatlab4
   7 0.11717500 fe80::cd82:6c83:ae0 ff02::1:3        LLMNR    90 Standard query 0x7d51  A itmmatlab4
   8 0.11717700 10.0.3.5            224.0.0.252      LLMNR    70 Standard query 0x7d51  A itmmatlab4
   9 0.13263000 10.0.3.5            10.255.255.255   NBNS     92 Name query NB WPAD<00>
  10 0.21514000 10.0.1.30           10.255.255.255   NBNS     92 Name query NB AMNSREIUOJY.RU<00>
  13 0.32018400 10.0.3.5            10.255.255.255   NBNS     92 Name query NB ITMMATLAB4<00>
  16 0.32932000 fe80::cd82:6c83:ae0 ff02::1:3        LLMNR    90 Standard query 0x8d5b  A itmmatlab4
  17 0.32932200 10.0.3.5            224.0.0.252      LLMNR    70 Standard query 0x8d5b  A itmmatlab4
  19 0.42920300 fe80::cd82:6c83:ae0 ff02::1:3        LLMNR    90 Standard query 0x8d5b  A itmmatlab4
  20 0.42920500 10.0.3.5            224.0.0.252      LLMNR    70 Standard query 0x8d5b  A itmmatlab4
  22 0.45895200 fe80::848e:a3b8:4b3 ff02::1:3        LLMNR    84 Standard query 0x96b5  A wpad
  23 0.45897600 10.0.3.8            224.0.0.252      LLMNR    64 Standard query 0x96b5  A wpad
  25 0.56354400 fe80::848e:a3b8:4b3 ff02::1:3        LLMNR    84 Standard query 0x96b5  A wpad
  26 0.56356900 10.0.3.8            224.0.0.252      LLMNR    64 Standard query 0x96b5  A wpad
  27 0.64044800 172.16.0.124        172.16.255.255   NBNS     92 Name query NB WPAD<00>
  34 0.76639200 10.0.3.8            10.255.255.255   NBNS     92 Name query NB WPAD<00>
  35 0.89702300 10.0.3.5            10.255.255.255   NBNS     92 Name query NB WPAD<00>
  47 1.52175100 10.0.8.11           10.255.255.255   NBNS     92 Name query NB ARMMF.ADOBE.COM<00>
  48 1.53067600 10.0.3.8            10.255.255.255   NBNS     92 Name query NB WPAD<00>
  50 1.62350100 10.0.1.50           10.255.255.255   NBNS     92 Name query NB MDA.NO-IP.ORG<00>
  51 1.66618400 fe80::cd82:6c83:ae0 ff02::1:3        LLMNR    84 Standard query 0x7cfb  A wpad
  52 1.66618600 10.0.3.5            224.0.0.252      LLMNR    64 Standard query 0x7cfb  A wpad
  55 1.77066000 fe80::cd82:6c83:ae0 ff02::1:3        LLMNR    84 Standard query 0x7cfb  A wpad
  56 1.77066200 10.0.3.5            224.0.0.252      LLMNR    64 Standard query 0x7cfb  A wpad
```

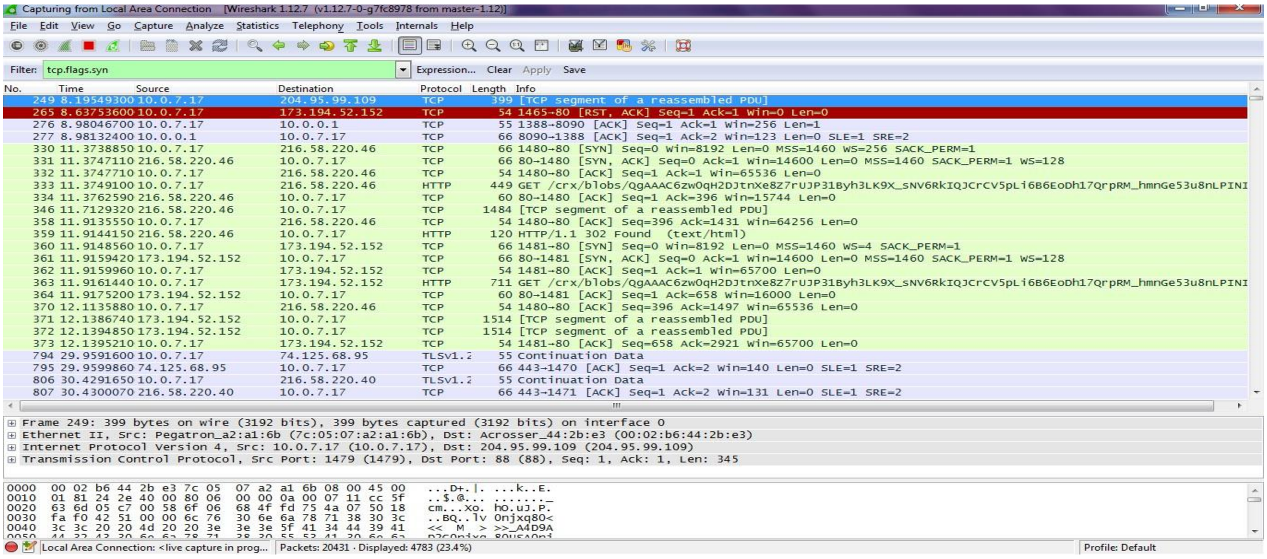
```
⊞ Frame 27: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
⊞ Ethernet II, Src: IntelCor_b1:98:2a (00:26:c7:b1:98:2a), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Internet Protocol Version 4, Src: 172.16.0.124 (172.16.0.124), Dst: 172.16.255.255 (172.16.255.255)
⊞ User Datagram Protocol, Src Port: 137 (137), Dst Port: 137 (137)
⊞ NetBIOS Name Service
```

```
0000  ff ff ff ff ff ff 00 26  c7 b1 98 2a 08 00 45 00   .......& ...*..E.
0010  00 4e 0f 2a 00 00 40 11  12 d9 ac 10 00 7c ac 10   .N.*..@. .....|..
0020  ff ff 00 89 00 89 00 3a  6f 0b f5 51 01 10 00 01   .......: o..Q....
0030  00 00 00 00 00 00 20 46  48 46 41 45 42 45 45 43   ...... F HFAEBEEC
0040  41 43 41 43 41 43 41 43  41 43 41 43 41 43 41 43   ACACACAC ACACACAC
0050  41 43 41 43 41 41 41 00  00 20 00 01              ACACAAA
```

File: "C:\Users\ADMINI~1\AppData\Local\T... Packets: 848 · Displayed: 432 (50.9%) · Dropped: 0 (0.0%) Profile: Default

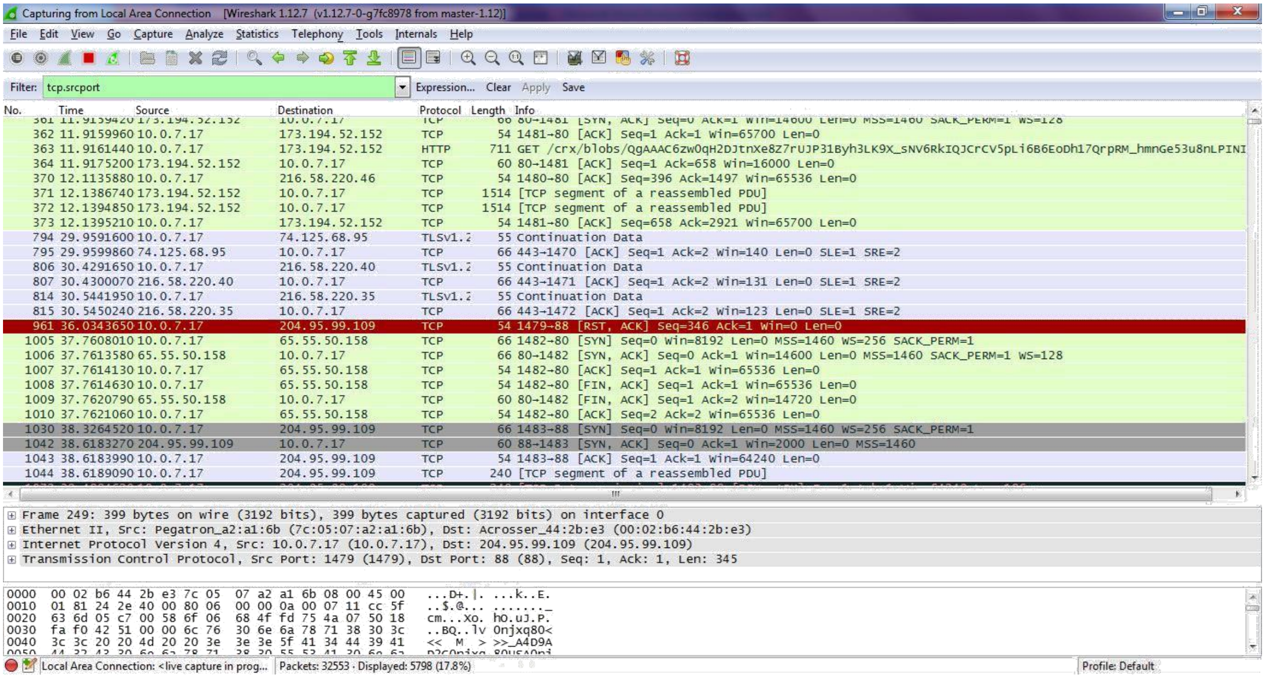
Filtering based on flags

Wireshark also has the ability to filter results based on TCP flags. For example, to display on those TCP packets that contain SYN flag, use the tcp.flags.syn filter. Here is an example: tcp.flags.syn



```
Capturing from Local Area Connection   [Wireshark 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)]
Filter: tcp.flags.syn

No.   Time        Source          Destination     Protocol Length Info
  249 8.19549300  10.0.7.17       204.95.99.109   TCP       399 [TCP segment of a reassembled PDU]
  265 8.63753600  10.0.7.17       173.194.52.152  TCP        54 1465→80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  276 8.98046700  10.0.7.17       10.0.0.1        TCP        55 1388→8090 [ACK] Seq=1 Ack=1 Win=256 Len=1
  277 8.98132400  10.0.0.1        10.0.7.17       TCP        66 8090→1388 [ACK] Seq=1 Ack=2 Win=123 Len=0 SLE=1 SRE=2
  330 11.3738850  10.0.7.17       216.58.220.46   TCP        66 1480→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
  331 11.3747110  216.58.220.46   10.0.7.17       TCP        66 80→1480 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=128
  332 11.3747710  10.0.7.17       216.58.220.46   TCP        54 1480→80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
  333 11.3749100  10.0.7.17       216.58.220.46   HTTP      449 GET /crx/blobs/QgAAAC6zw0qH2DJtnXe8Z7rUJP31Byh3LK9X_sNV6RkIQJCrCV5pLi6B6EoDh17QrpRM_hmnGe53u8nLPINI
  334 11.3762590  216.58.220.46   10.0.7.17       TCP        60 80→1480 [ACK] Seq=1 Ack=396 Win=15744 Len=0
  346 11.7129320  216.58.220.46   10.0.7.17       TCP      1484 [TCP segment of a reassembled PDU]
  358 11.9135550  10.0.7.17       216.58.220.46   TCP        54 1480→80 [ACK] Seq=396 Ack=1431 Win=64256 Len=0
  359 11.9144150  216.58.220.46   10.0.7.17       HTTP      120 HTTP/1.1 302 Found  (text/html)
  360 11.9148560  10.0.7.17       173.194.52.152  TCP        66 1481→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
  361 11.9159420  173.194.52.152  10.0.7.17       TCP        66 80→1481 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=128
  362 11.9159960  10.0.7.17       173.194.52.152  TCP        54 1481→80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
  363 11.9161440  10.0.7.17       173.194.52.152  HTTP      711 GET /crx/blobs/QgAAAC6zw0qH2DJtnXe8Z7rUJP31Byh3LK9X_sNV6RkIQJCrCV5pLi6B6EoDh17QrpRM_hmnGe53u8nLPINI
  364 11.9175200  173.194.52.152  10.0.7.17       TCP        60 80→1481 [ACK] Seq=1 Ack=658 Win=16000 Len=0
  370 12.1135880  10.0.7.17       216.58.220.46   TCP        54 1480→80 [ACK] Seq=396 Ack=1497 Win=65536 Len=0
  371 12.1386740  173.194.52.152  10.0.7.17       TCP      1514 [TCP segment of a reassembled PDU]
  372 12.1394850  173.194.52.152  10.0.7.17       TCP      1514 [TCP segment of a reassembled PDU]
  373 12.1395210  10.0.7.17       173.194.52.152  TCP        54 1481→80 [ACK] Seq=658 Ack=2921 Win=65700 Len=0
  794 29.9591600  10.0.7.17       74.125.68.95    TLSv1.2    55 Continuation Data
  795 29.9599860  74.125.68.95    10.0.7.17       TCP        66 443→1470 [ACK] Seq=1 Ack=2 Win=140 Len=0 SLE=1 SRE=2
  806 30.4291650  10.0.7.17       216.58.220.40   TLSv1.2    55 Continuation Data
  807 30.4300070  216.58.220.40   10.0.7.17       TCP        66 443→1471 [ACK] Seq=1 Ack=2 Win=131 Len=0 SLE=1 SRE=2

⊞ Frame 249: 399 bytes on wire (3192 bits), 399 bytes captured (3192 bits) on interface 0
⊞ Ethernet II, Src: Pegatron_a2:a1:6b (7c:05:07:a2:a1:6b), Dst: Acrosser_44:2b:e3 (00:02:b6:44:2b:e3)
⊞ Internet Protocol Version 4, Src: 10.0.7.17 (10.0.7.17), Dst: 204.95.99.109 (204.95.99.109)
⊞ Transmission Control Protocol, Src Port: 1479 (1479), Dst Port: 88 (88), Seq: 1, Ack: 1, Len: 345

0000  00 02 b6 44 2b e3 7c 05  07 a2 a1 6b 08 00 45 00   ...D+.|. ...k..E.
0010  01 81 24 2e 40 00 80 06  00 00 0a 00 07 11 cc 5f   ..$.@... ......._
0020  63 6d 05 c7 00 58 6f 06  68 4f fd 75 4a 07 50 18   cm...Xo. hO.uJ.P.
0030  fa f0 42 51 00 00 6c 76  30 6e 6a 78 71 38 30 3c   ..BQ..lv 0njxq80<
0040  3c 3c 20 20 4d 20 20 3e  3e 3e 5f 41 34 44 39 41   <<  M  > >>_A4D9A

Local Area Connection: <live capture in prog...   Packets: 20431 · Displayed: 4783 (23.4%)   Profile: Default
```

TCP Source Port tcp.srcport

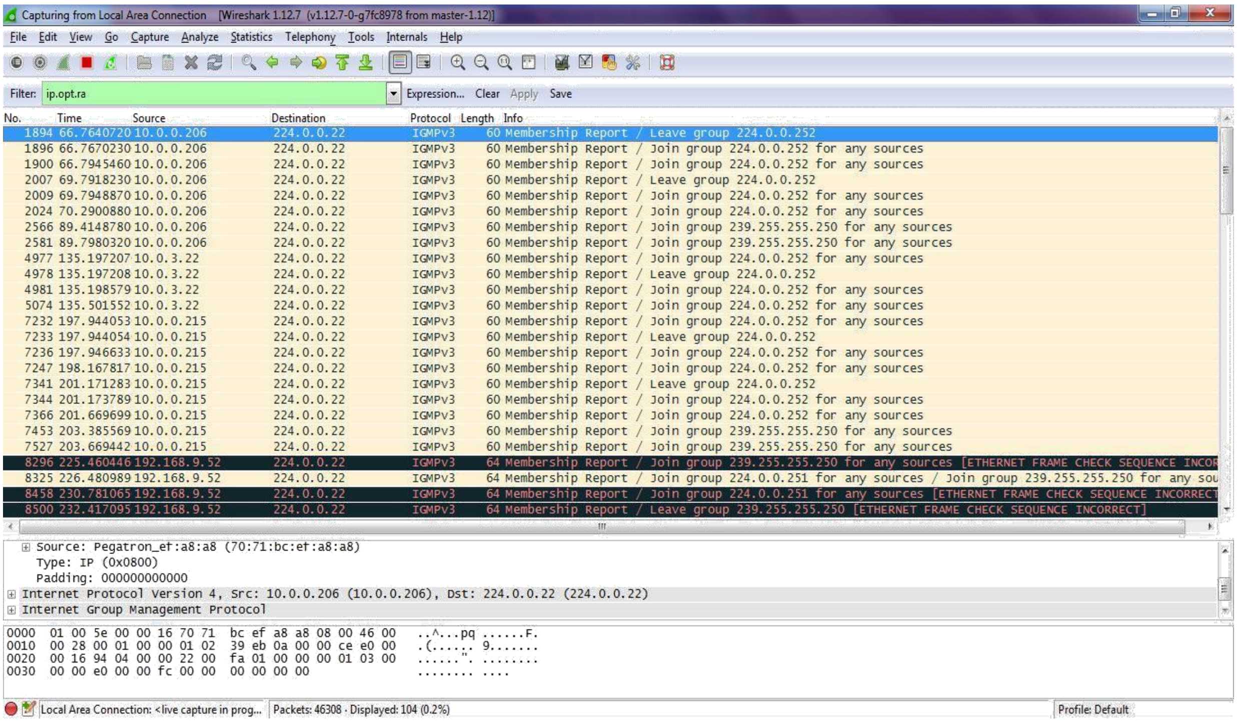


```
Capturing from Local Area Connection   [Wireshark 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)]
Filter: tcp.srcport

No.   Time        Source          Destination     Protocol Length Info
  361 11.9159420  173.194.52.152  10.0.7.17       TCP        66 80→1481 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=128
  362 11.9159960  10.0.7.17       173.194.52.152  TCP        54 1481→80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
  363 11.9161440  10.0.7.17       173.194.52.152  HTTP      711 GET /crx/blobs/QgAAAC6zw0qH2DJtnXe8Z7rUJP31Byh3LK9X_sNV6RkIQJCrCV5pLi6B6EoDh17QrpRM_hmnGe53u8nLPINI
  364 11.9175200  173.194.52.152  10.0.7.17       TCP        60 80→1481 [ACK] Seq=1 Ack=658 Win=16000 Len=0
  370 12.1135880  10.0.7.17       216.58.220.46   TCP        54 1480→80 [ACK] Seq=396 Ack=1497 Win=65536 Len=0
  371 12.1386740  173.194.52.152  10.0.7.17       TCP      1514 [TCP segment of a reassembled PDU]
  372 12.1394850  173.194.52.152  10.0.7.17       TCP      1514 [TCP segment of a reassembled PDU]
  373 12.1395210  10.0.7.17       173.194.52.152  TCP        54 1481→80 [ACK] Seq=658 Ack=2921 Win=65700 Len=0
  794 29.9591600  10.0.7.17       74.125.68.95    TLSv1.2    55 Continuation Data
  795 29.9599860  74.125.68.95    10.0.7.17       TCP        66 443→1470 [ACK] Seq=1 Ack=2 Win=140 Len=0 SLE=1 SRE=2
  806 30.4291650  10.0.7.17       216.58.220.40   TLSv1.2    55 Continuation Data
  807 30.4300070  216.58.220.40   10.0.7.17       TCP        66 443→1471 [ACK] Seq=1 Ack=2 Win=131 Len=0 SLE=1 SRE=2
  814 30.5441950  10.0.7.17       216.58.220.35   TLSv1.2    55 Continuation Data
  815 30.5450240  216.58.220.35   10.0.7.17       TCP        66 443→1472 [ACK] Seq=1 Ack=2 Win=123 Len=0 SLE=1 SRE=2
  961 36.0343650  10.0.7.17       204.95.99.109   TCP        54 1479→88 [RST, ACK] Seq=346 Ack=1 Win=0 Len=0
 1005 37.7608010  10.0.7.17       65.55.50.158    TCP        66 1482→80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 1006 37.7613580  65.55.50.158    10.0.7.17       TCP        66 80→1482 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK_PERM=1 WS=128
 1007 37.7614130  10.0.7.17       65.55.50.158    TCP        54 1482→80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
 1008 37.7614630  10.0.7.17       65.55.50.158    TCP        54 1482→80 [FIN, ACK] Seq=1 Ack=1 Win=65536 Len=0
 1009 37.7620790  65.55.50.158    10.0.7.17       TCP        60 80→1482 [FIN, ACK] Seq=1 Ack=2 Win=14720 Len=0
 1010 37.7621060  10.0.7.17       65.55.50.158    TCP        54 1482→80 [ACK] Seq=2 Ack=2 Win=65536 Len=0
 1030 38.3264520  10.0.7.17       204.95.99.109   TCP        66 1483→88 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
 1042 38.6183270  204.95.99.109   10.0.7.17       TCP        60 88→1483 [SYN, ACK] Seq=0 Ack=1 Win=2000 Len=0 MSS=1460
 1043 38.6183990  10.0.7.17       204.95.99.109   TCP        54 1483→88 [ACK] Seq=1 Ack=1 Win=64240 Len=0
 1044 38.6189090  10.0.7.17       204.95.99.109   TCP       240 [TCP segment of a reassembled PDU]

⊞ Frame 249: 399 bytes on wire (3192 bits), 399 bytes captured (3192 bits) on interface 0
⊞ Ethernet II, Src: Pegatron_a2:a1:6b (7c:05:07:a2:a1:6b), Dst: Acrosser_44:2b:e3 (00:02:b6:44:2b:e3)
⊞ Internet Protocol Version 4, Src: 10.0.7.17 (10.0.7.17), Dst: 204.95.99.109 (204.95.99.109)
⊞ Transmission Control Protocol, Src Port: 1479 (1479), Dst Port: 88 (88), Seq: 1, Ack: 1, Len: 345

0000  00 02 b6 44 2b e3 7c 05  07 a2 a1 6b 08 00 45 00   ...D+.|. ...k..E.
0010  01 81 24 2e 40 00 80 06  00 00 0a 00 07 11 cc 5f   ..$.@... ......._
0020  63 6d 05 c7 00 58 6f 06  68 4f fd 75 4a 07 50 18   cm...Xo. hO.uJ.P.
0030  fa f0 42 51 00 00 6c 76  30 6e 6a 78 71 38 30 3c   ..BQ..lv 0njxq80<
0040  3c 3c 20 20 4d 20 20 3e  3e 3e 5f 41 34 44 39 41   <<  M  > >>_A4D9A

Local Area Connection: <live capture in prog...   Packets: 32553 · Displayed: 5798 (17.8%)   Profile: Default
```

Router Alert

ip.opt.ra

Unsigned integer, 2 bytes 1.8.0 to 1.12.6


Capturing from Local Area Connection [Wireshark 1.12.7 (v1.12.7-0-g7fc8978 from master-1.12)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save
Profile: Default

Filter: ip.opt.ra

```
No.     Time        Source          Destination     Protocol Length Info
   1894 66.7640720 10.0.0.206      224.0.0.22      IGMPv3     60 Membership Report / Leave group 224.0.0.252
   1896 66.7670230 10.0.0.206      224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   1900 66.7945460 10.0.0.206      224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   2007 69.7918230 10.0.0.206      224.0.0.22      IGMPv3     60 Membership Report / Leave group 224.0.0.252
   2009 69.7948870 10.0.0.206      224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   2024 70.2900880 10.0.0.206      224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   2566 89.4148780 10.0.0.206      224.0.0.22      IGMPv3     60 Membership Report / Join group 239.255.255.250 for any sources
   2581 89.7980320 10.0.0.206      224.0.0.22      IGMPv3     60 Membership Report / Join group 239.255.255.250 for any sources
   4977 135.197207 10.0.3.22       224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   4978 135.197208 10.0.3.22       224.0.0.22      IGMPv3     60 Membership Report / Leave group 224.0.0.252
   4981 135.198579 10.0.3.22       224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   5074 135.501552 10.0.3.22       224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   7232 197.944053 10.0.0.215      224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   7233 197.944054 10.0.0.215      224.0.0.22      IGMPv3     60 Membership Report / Leave group 224.0.0.252
   7236 197.946633 10.0.0.215      224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   7247 198.167817 10.0.0.215      224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   7341 201.171283 10.0.0.215      224.0.0.22      IGMPv3     60 Membership Report / Leave group 224.0.0.252
   7344 201.173789 10.0.0.215      224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   7366 201.669699 10.0.0.215      224.0.0.22      IGMPv3     60 Membership Report / Join group 224.0.0.252 for any sources
   7453 203.385569 10.0.0.215      224.0.0.22      IGMPv3     60 Membership Report / Join group 239.255.255.250 for any sources
   7527 203.669442 10.0.0.215      224.0.0.22      IGMPv3     60 Membership Report / Join group 239.255.255.250 for any sources
   8296 225.460446 192.168.9.52    224.0.0.22      IGMPv3     64 Membership Report / Join group 239.255.255.250 for any sources [ETHERNET FRAME CHECK SEQUENCE INCOR
   8325 226.480989 192.168.9.52    224.0.0.22      IGMPv3     64 Membership Report / Join group 224.0.0.251 for any sources / Join group 239.255.255.250 for any sou
   8458 230.781065 192.168.9.52    224.0.0.22      IGMPv3     64 Membership Report / Join group 224.0.0.251 for any sources [ETHERNET FRAME CHECK SEQUENCE INCORRECT
   8500 232.417095 192.168.9.52    224.0.0.22      IGMPv3     64 Membership Report / Leave group 239.255.255.250 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
```

```
  ⊞ Source: Pegatron_ef:a8:a8 (70:71:bc:ef:a8:a8)
    Type: IP (0x0800)
    Padding: 000000000000
⊞ Internet Protocol Version 4, Src: 10.0.0.206 (10.0.0.206), Dst: 224.0.0.22 (224.0.0.22)
⊞ Internet Group Management Protocol
```

```
0000  01 00 5e 00 00 16 70 71  bc ef a8 a8 08 00 46 00   ..^...pq ......F.
0010  00 28 00 01 00 00 01 02  39 eb 0a 00 00 ce e0 00   .(...... 9.......
0020  00 16 94 04 00 00 22 00  fa 01 00 00 00 01 03 00   ......". ........
0030  00 00 e0 00 00 fc 00 00  00 00 00 00               ........ ....
```

Local Area Connection: <live capture in prog... Packets: 46308 · Displayed: 104 (0.2%)