DIGITAL SIGNAL PROCESSING LAB (EE-429-F)

LAB MANUAL

VII SEMESTER

Department Of Electronics & Communication Engg
Dronacharya College Of Engineering
Khentawas, Gurgaon – 123506

LIST OF EXPERIMENTS

EXPERIMENT NO. 1(a)

AIM: - TO ILLUSTRATE THE SIMPLE MATHEMATICAL EXPRESSIONS IN MATLAB.

PROGRAM:-

1.
```
clc;
clear all;

close all;

x=2^5

y=x-1

z=x/y
```

OUTPUT :-

```
x= 32

y= 31

z=1.0323
```

2.
```
clc;
clear all;

close all;

x=3*sqrt(5)-1

y=(sqrt(5)+1^2

z=x/y-1
```

OUTPUT :-

```
x= 5.7082

y= 10.4721

z=-0.4549
```

3.
```
clc;
clear all;
```

```
close all;

r= pi^(1/3)-1

area=pi*r^2
```

OUTPUT :-

```
r= 0.4646

area=0.6781
```

4.
```
clc;
clear all;

close all;

r= sqrt(163)

t= exp(pi*r)
```

OUTPUT :-

```
r= 12.7671

t= 2.6254e+017
```

5.
```
clc;
clear all;

close all;

x=(sin(pi/6))^2

y=(cos(pi/6))^2

t=x+y
```

OUTPUT :-

```
x= 0.2500

y= 0.7500

t=1
```

6.
```
clc;
```

```
clear all;

close all;

t=pi/4

x=exp(i*t)
```

OUTPUT :-

```
t= 0.78.54

x= 0.7071+0.7071i
```

7.
```
clc;
clear all;

close all;

t= exp (pi/(4*i))
```

OUTPUT :-

```
t= 32

y= 31

z=0.7071+0.7071i
```

8.
```
clc;
clear all;

close all;

t= 1:1:10

z= sin(t.^2)/ (t.^2)
```

OUTPUT :-

```
t= 1    2    3    4    5    6    7    8    9    10

z= -0.0052
```

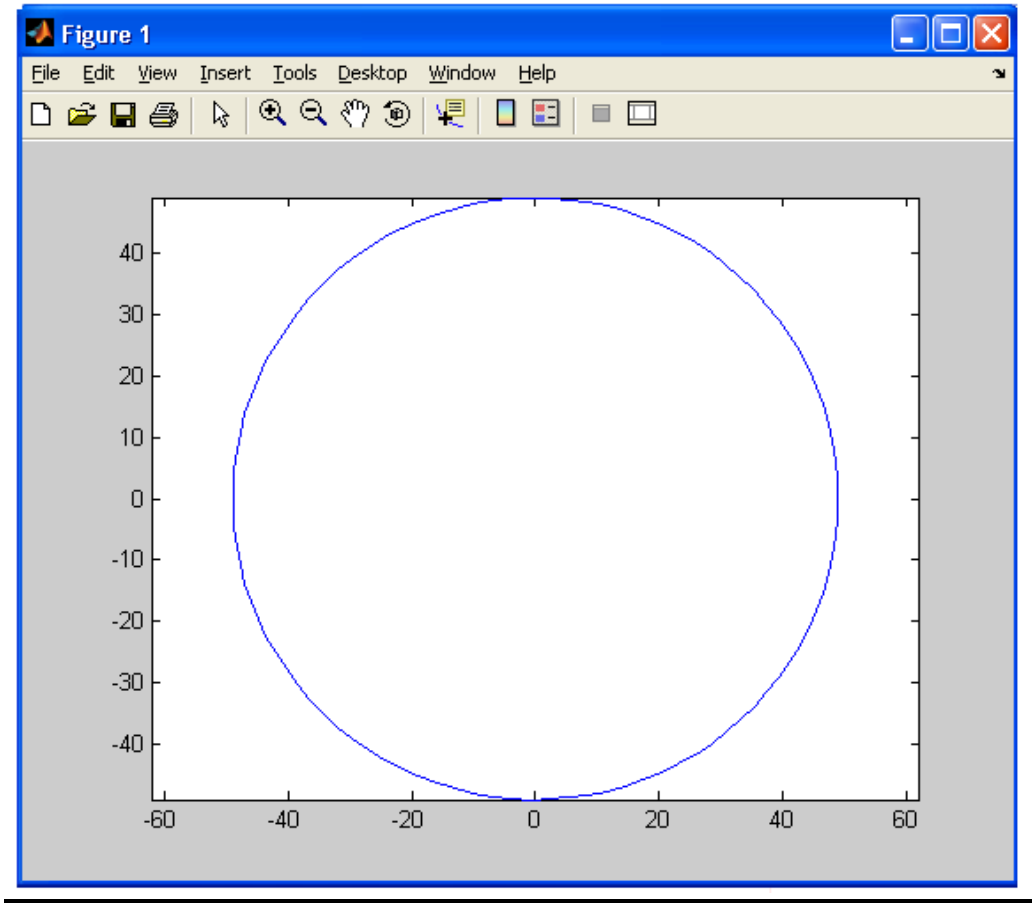
EXPERIMENT NO. 1(b)

AIM:- To design a program to draw a unit circle

PROGRAM:-

```
clc;
clear all;
close all;
r=('1');
theta=linespace(0,2*p,100);
x=r*cos(theta);
y=r*sin(theta);
plot(x,y);
xlable('x');
ylabel('y');
title('circle');
```

OUTPUT:-



QUESTIONS/ANSWERS

Q1.What is a MATLAB?.

A1. **MATLAB** (**mat**rix **lab**oratory) is a numerical computing environment andfourth-generation programming language

Q2 What are the various functions of MATLAB?

A2 MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran Star and ring are the two examples of hybrid topology

Q3. What is the difference between an array and a vector?

A3. A *vector* refers to a one dimensional (1×*N* or *N*×1) matrix, commonly referred to as an array in other programming languages. A *matrix* generally refers to a 2-dimensional array, i.e. an *m*×*n* array where *m* and *n* are greater than 1.

Q4. What are the various windows in MATLAB?

A4 Command window, Editor window, Launch pad Window

Q5 What are variables?

A5. Variables are assigned numerical values by typing the expression directly, for example, typing

a = 1+2 yields: a = 3

The answer will not be displayed when a semicolon is put at the end of an expression, for example type **a = 1+2;**

Q6 What are the various arithmetic operators used in MATLAB?

A6. MATLAB utilizes the following arithmetic operators:

+ addition
- subtraction
* multiplication
/ division
^ power operator
' transpose

Q7. What are the various special type of matrices used in MATLAB?

A7.

null matrix:	**M = [ ];**
nxm matrix of zeros:	**M = zeros(n,m);**
nxm matrix of ones:	**M = ones(n,m);**
nxn identity matrix:	**M = eye(n);**

Q8. Is MATLAB a case sensitive?

A8. Yes. "a" and "A" are two different names.

Q9. How we define comment statements in MATLAB?

A9. Comment statements are preceded by a "%"..

Q10.What are M-files ?

A10. M-files are macros of MATLAB commands that are stored as ordinary text files with the extension "m", that is *filename*.m. An M-file can be either a function with input and output variables or a list of commands. All of the MATLAB examples in this textbook are contained in M-files that are available at the Math Works ftp site, ftp.mathworks.com in the directory pub/books/heck.

EXPERIMENT NO. 2

AIM:- To present basic signals(unit step, unit impulse, ramp, exponent, sine and cosine)

PROGRAM:-

```
** for unit step, unit impulse, ramp and exponent
clc;
clear all;
close all;
n=[0:0.5:20];
x=input('enter value of x');
y=exp(x*n);
subplot(2,2,1);
stem(n,y);
xlabel ('x');
ylabel ('exponent');
title('expo1');
%clc;
%clear all;
%close all;
n=[0:0.5:20];
a=input('enter value of a');
y=(a).^n;
subplot(2,2,2);
stem(n,y);
xlabel('a');
ylabel('exponent');
title('expo2');
%clc;
%clear all;
%close all;
n=[0:0.5:20];
subplot(2,2,3);
stem(n,n);
xlabel('x');
ylabel('y');
title('unit ramp');
```

```
%clc;
%clear all;
%close all;
t=[-3:1:2];
n1=[zeros(1,3),ones(1,1),zeros(1,2)];
subplot(2,2,4);
stem(t,n1);
xlabel('x');
ylabel('y');
title('unit impulse');

**for sine and cosine
clc;
clear all;
close all;
t=[0:0.1:pi];
y=cos(2*pi*t);
subplot(2,2,1);
stem(t,y);
xlabel('t');
ylabel('coz');
title('cos(2pit)');
%clc;
%clear all;
%close all;
t=[0:0.1:pi];
y=sin(2*pi*t);
subplot(2,2,2);
stem(t,y);
xlabel('t');
ylabel('sin');
title('sin(2pit)');
```

OUTPUT:-

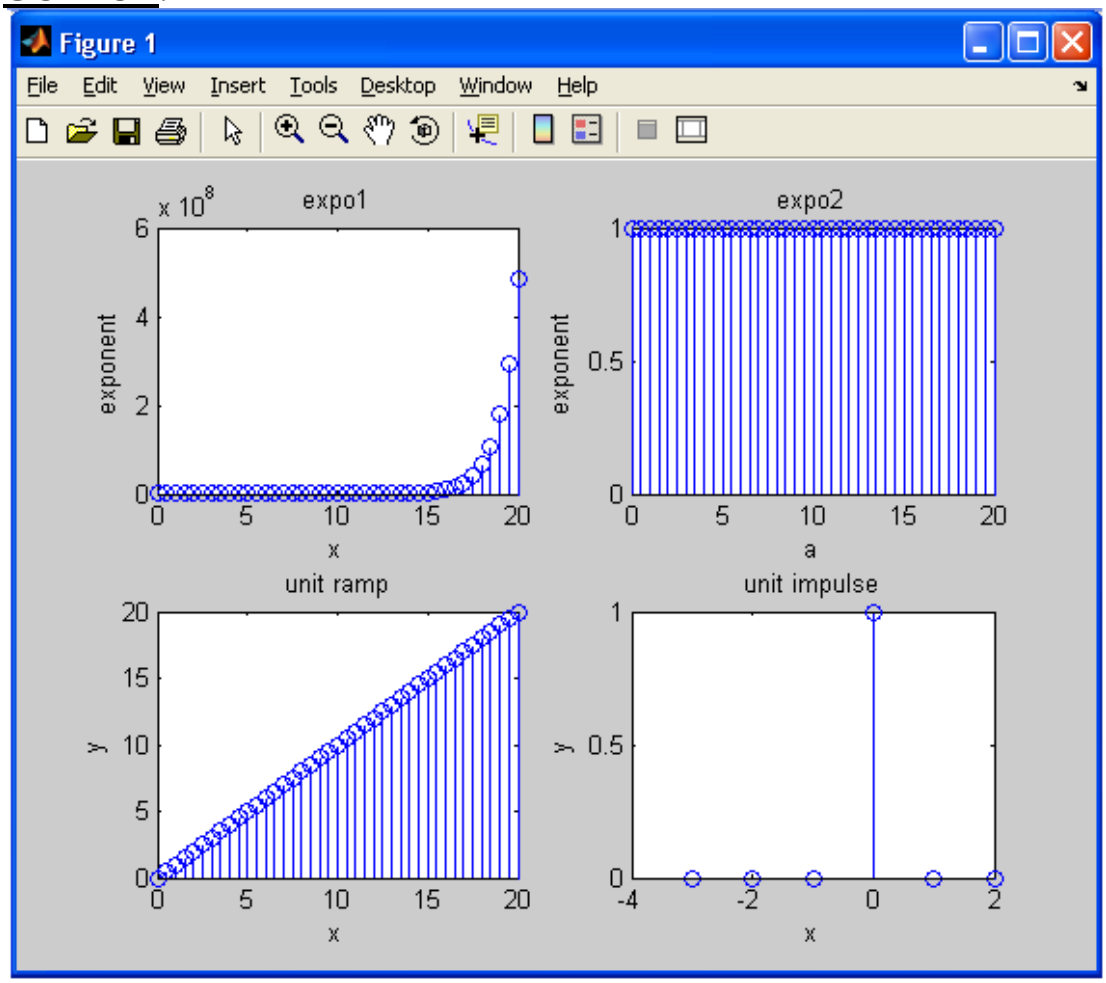


Figure 1
File Edit View Insert Tools Desktop Window Help
expo1
x 10^8
exponent
x
expo2
exponent
a
unit ramp
y
x
unit impulse
y
x

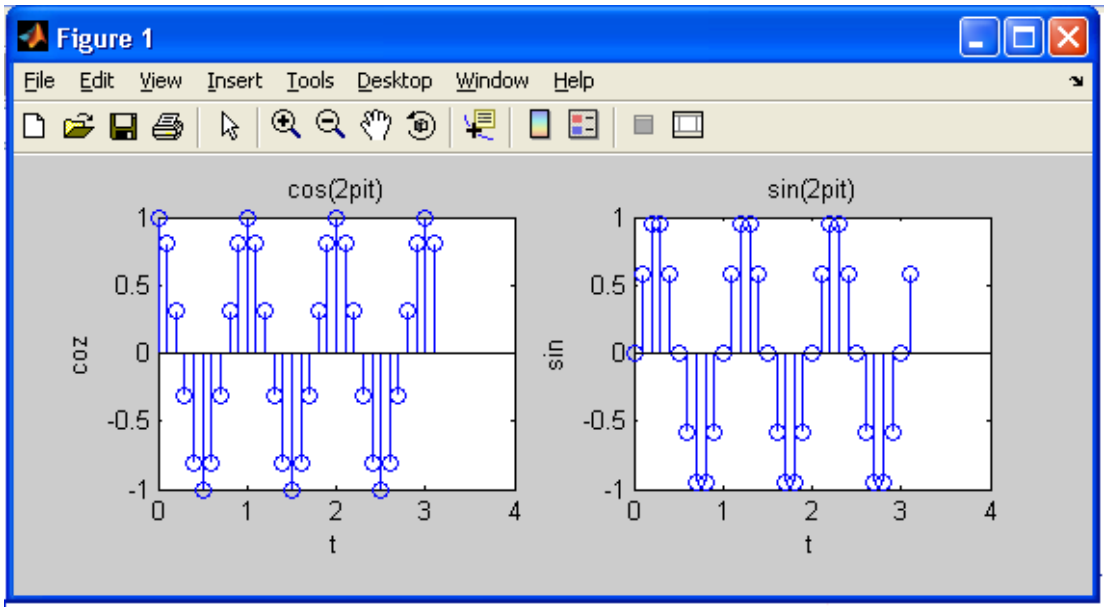


Figure 1
File Edit View Insert Tools Desktop Window Help
cos(2pit)
coz
t
sin(2pit)
sin
t

QUESTIONS/ANSWERS

Q1. Which command is used to draw a continuous waveform?

A2 The command most often used for plotting is plot, which creates linear plots of vectors and matrices; plot(t,y) plots the vector t on the x-axis versus vector y on the y-axis

Q2 Which command is used to draw a discrete waveform?

A2 For discrete-time signals, use the command stem which plots each point with a small open circle and a straight line. To plot y[k] versus k, type stem(k,y).

Q3. Which command is used to plot two or more graphs on the same set of axes?

A3.To plot two or more graphs on the same set of axes, use the command plot (t1,y1,t2,y2),which plots y1 versus t1 and y2 versus t2.

Q4. Which command is used to label the axes?

A4 To label your axes , type e.g.

```
xlabel('time (sec)')
ylabel('step response')
```

Q5 Which command is used to give a title name to the plot?

A5 To give the plot a title, type

```
Title ('My Plot')
```

Q6 Which command is used to add a grid to the plot?

A6 To add a grid to your plot to make it easier to read. Type

grid

Q7. Which command is used to plot more than one graph on the screen?

A7. To plot more than one graph on the screen, use the command subplot(mnp) which partitions the screen into an m x n grid where p determines the position of the particular graph counting the upper left corner as p=1. For example,

```
subplot(2,1,1)
```

Q8.For autoscaling of the axes which command is used?

A8: The auto scaling of the axes can be done by using the axis command after the plotting command:

axis([xmin xmax ymin ymax]);

where xmin, xmax, ymin, and ymax are numbers corresponding to the limits you desire for the axes. To return to the automatic scaling, simply type axis.

Q9. How the colour of plot can be changed?

A9 There are options on the line type and the color of the plot which are obtained using plot (t,y,'option'). The linetype options are '-' solid line (default), '--' dashed line, '-.' dot dash line, ':' dotted line. The points in y can be left unconnected and delineated by a variety of symbols: + . * o x. The following colors are available options:

r	red
b	blue
g	green
w	white
k	black

Q10.Which command is used to make the axis equal?

A10: axis (‘equal’)

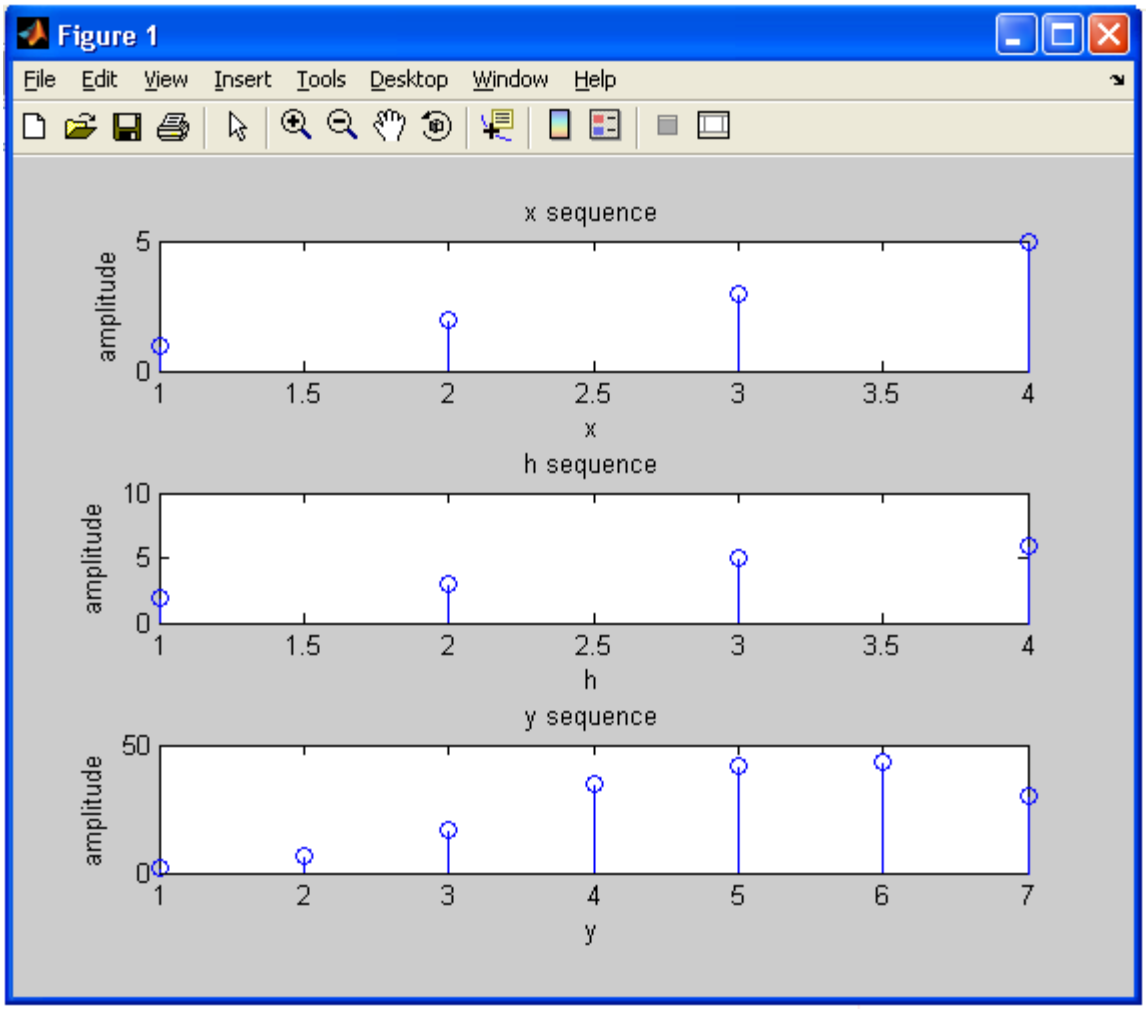
EXPERIMENT NO. 3

AIM:- To develop a program for discrete convolution.

PROGRAM:-

```
clc,
clear all;
close all;
x=[1 2 3 5];
h=[2 3 5 6];
y=conv(x,h);
subplot(3,1,1);
stem(x);
xlabel('x');
ylabel('amplitude');
title('x sequence');
subplot(3,1,2);
stem(h);
xlabel('h');
ylabel('amplitude');
title('h sequence');
subplot(3,1,3);
stem(y);
xlabel('y');
ylabel('amplitude');
title('y sequence');
```

OUTPUT:-



QUESTIONS/ANSWERS

Q1.What is convolution?

A1 In mathematics and, in particular, functional analysis, **convolution** is a mathematical operation on two functions f and g, producing a third function that is typically viewed as a modified version of one of the original functions, giving the area overlap between the two functions as a function of the amount that one of the original functions is translated.

Q2. What are the applications of convolution?

A2 It has applications that include probability, statistics, computer vision, image and signal processing, electrical engineering, and differential equations.

Q3. What is the difference between circular convolution and discrete convolution?

A3 the circular convolution can be defined for periodic functions (that is, functions on the circle), and the discrete convolution can be defined for functions on the set of integers.

Q4. What is deconvolution?

A4 Computing the inverse of the convolution operation is known as deconvolution.
N two sequences

Q5 What is the symbol for convolution?

A5. The convolution of f and g is written f∗g, using an asterisk or star. It is defined as the integral of the product of the two functions after one is reversed and shifted.

Q6 Which command is used to find convolution between two sequences?

A6.If x & h are two sequences then the command used to find convolution between these sequences can be represented by Conv(x,h).

Q7. What are the various special type of matrices used in MATLAB?

A7.

null matrix:	**M = [ ];**
nxm matrix of zeros:	**M = zeros(n,m);**
nxm matrix of ones:	**M = ones(n,m);**
nxn identity matrix:	**M = eye(n);**

Q8. Is convolution obeys the algebraic properties?

A8. Yes. It satisfies associative , commutative , distributive algebraic properties.

Q9. What is a convolution matrix?

A9. A convolution matrix is a matrix, formed from a vector, whose inner product with another vector is the convolution of the two vectors.

Q10 Generate a simple convolution matrix.

A10. If

```
h = [1 2 3 2 1];
convmtx(h,7);
```

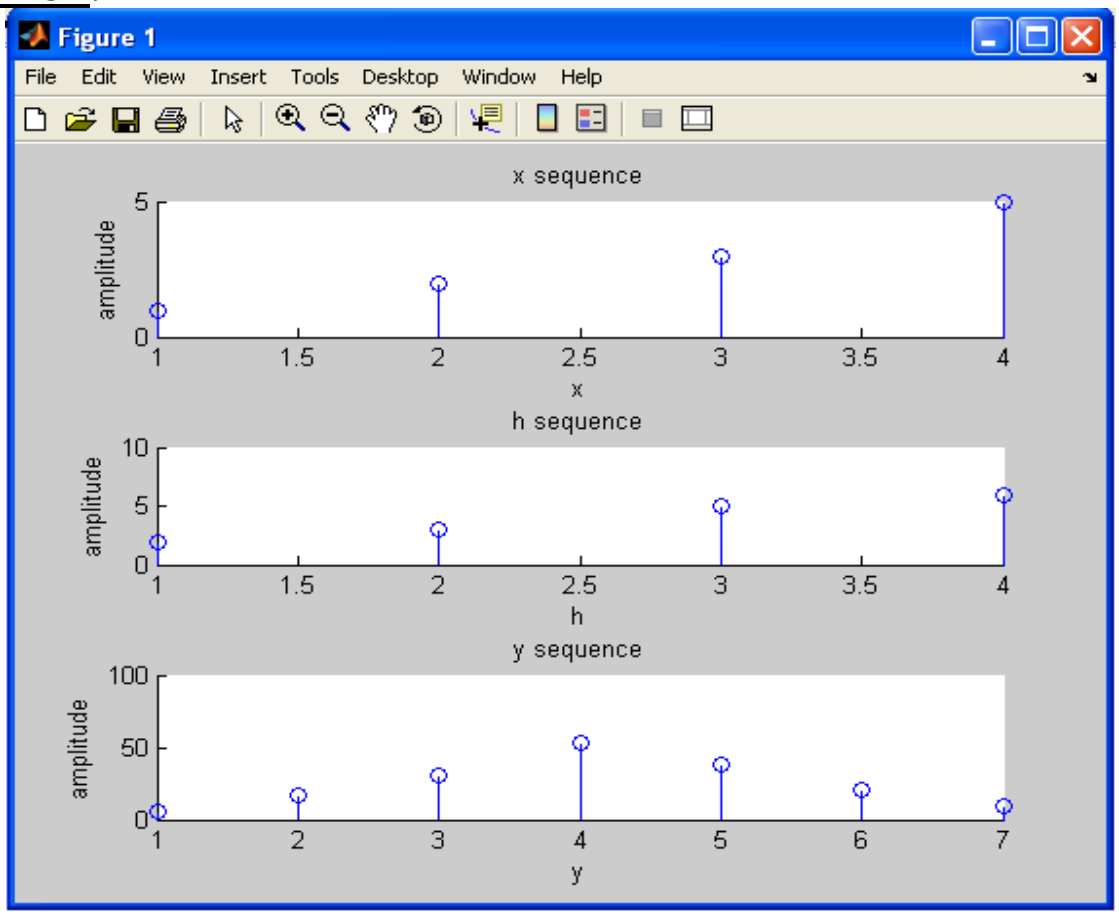
EXPERIMENT NO. 4

AIM:- To develop a program for discrete correlation

PROGRAM:-

```
clc,
clear all;
close all;
x=[1 2 3 5];
h=[2 3 5 6];
y=xcorr(x,h);
subplot(3,1,1);
stem(x);
xlabel('x');
ylabel('amplitude');
title('x sequence');
subplot(3,1,2);
stem(h);
xlabel('h');
ylabel('amplitude');
title('h sequence');
subplot(3,1,3);
stem(y);
xlabel('y');
ylabel('amplitude');
title('y sequence');
```

OUTPUT:-



QUESTIONS/ANSWERS

Q1.What is correlation?

A1 *Correlation* quantifies the strength of a linear relationship between two variables. When there is no correlation between two variables, then there is no tendency for the values of the variables to increase or decrease in tandem. Two variables that are uncorrelated are not necessarily independent, however, because they might have a nonlinear relationship.

Q2. What are the applications of correlation?

A2 It has applications that computes and plots the sample autocorrelation function (ACF) of a univariate, stochastic time series .

Q3. What is auto-correlation?

A3 The autocorrelation function of a random signal describes the general dependence of the values of the samples at one time on the values of the samples at another time.

Q4. What is cross-correlation?

A4 The cross correlation function however measures the dependence of the values of one signal on another signal.

Q5 Which command is used to find the correlation coefficients

A5.r=corr2(A,B) computes the correlation coefficient between A and B, where A and B are matrices or vectors of the same size.

Q6 Which command is used to find correlation between two sequences?

A6 If x & y are two sequences then

```
c=xcorr(x,y)
```

Q7. What command is used to find cross-correlation of two matrices?

A7:C = xcorr2(A,B) returns the cross-correlation of matrices A and B with no scaling. xcorr2 is the two-dimensional version of xcorr. It has its maximum value when the two matrices are aligned so that they are shaped as similarly as possible.

Q8. Is convolution obeys the algebraic properties?

A8. Yes. It satisfies associative , commutative , distributive algebraic properties.

Q9. What is a convolution matrix?

A9. A convolution matrix is a matrix, formed from a vector, whose inner product with another vector is the convolution of the two vectors.

Q10 Generate a simple convolution matrix.

A10. If

```
h = [1 2 3 2 1];
convmtx(h,7);
```

EXPERIMENT NO. 5

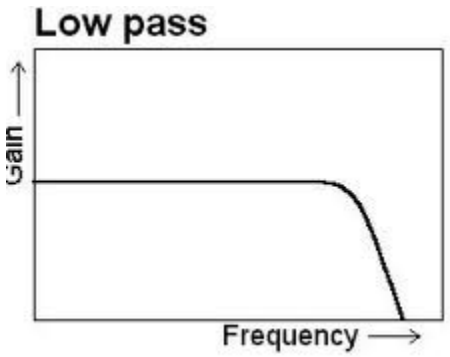
AIM : To design Infinite Impulse Response (low pass filter) filter with cut of frequency of 4000hz .

PROGRAM :

```
#include "dsk6713.h"                          //this file is added to initialize the DSK6713
#include "dsk6713_aic23.h"
Uint32 fs = DSK6713_AIC23_FREQ_8KHZ;  // set the sampling frequency, Different sampling
frequencies
                                                    supported by AIC23 codec are 8, 16, 24, 32,
44.1, 48, and
                                                    96 kHz.

// FILTER COFFICIENTS IS CALCULATED BY MATLAB


float fc [  ]={
                2.338110787e-019,6.936318823e-005,-0.0003181171778,0.0008399875951,
               - 0.001779771759,0.003340889933,-0.005792469252,  0.00948059652,-
0.01485389285,
                0.02252536267,-0.03342207149,  0.04916161299, -0.07310581207,
0.1139752045,
              -  0.2039434612, 0.6338219047,   0.6338219047,  -0.2039434612,   0.1139752045,
               - 0.07310581207, 0.04916161299, -0.03342207149,  0.02252536267, -
0.01485389285,
                0.00948059652, -0.005792469252, 0.003340889933,-
0.001779771759,0.0008399875951,
               - 0.0003181171778, 6.936318823e-005,2.338110787e-019
                 };

static short in_buffer[18] ;
void main( )
{

          comm_intr();                    // ISR function is called, using the given command
          while(1);                       // program execution halts and it starts listening for the
                                                  interrupt which occur at  every  sampling
period Ts.
}
interrupt void c_int11( )                  // ISR call, At each Interrupt, program execution goes
                                                to the interrupt service routine
{
        Uint32 indata;                    //variable declaration
        int i=0;
```

```
        signed int output=0;

        indata = input_sample();                //newest input @ top of buffer
        in_buffer[0] = indata;                  //new input at buffer[0]

        for(i=17;i>=0;i--)
        in_buffer[i] = in_buffer[i-1];          //shuffle the buffer

        for(i=0;i<18;i++)
        output = output + fc[i] * in_buffer[i];

        output_sample(output);                  //output filter,the value in the buffer yn indexed by
                                                  the variable  loop is written on to the codec.
}
```

OUTPUT :



QUESTION/ANSWER

Q1:-What are the advantages of IIR filters (compared to FIR filters)?

A1:-IIR filters can achieve a given filtering characteristic using less memory and calculations than a similar FIR filter

Q2:- What are the disadvantages of IIR filters (compared to FIR filters)?

A2:- They are more susceptable to problems of finite-length arithmetic, such as noise generated by calculations, and limit cycles. (This is a direct consequence of feedback: when the output isn't computed perfectly and is fed back, the imperfection can compound.)

- They are harder (slower) to implement using fixed-point arithmetic.

Q3:- Why is the impulse response "infinite"?

A3The impulse response is "infinite" because there is feedback in the filter; if you put in an impulse (a single "1" sample followed by many "0" samples), an infinite number of non-zero values will come out (theoretically).

Q4:- What are IIR filters? What does "IIR" mean?

IIR filters are one of two primary types of digital filters used in Digital Signal Processing (DSP) applications (the other type being FIR). "IIR" means "Infinite Impulse Response".

Q5:- Why is the impulse response "infinite"?

The impulse response is "infinite" because there is feedback in the filter; if you put in an impulse (a single "1" sample followed by many "0" samples), an infinite number of non-zero values will come out (theoretically).

Q6:- Give one advantage of digital filter over analog filter?

A1:- Digital filter can have characteristics which are not possible with analog filter

Q7:-What is the limitation of approximation method

A7:-This technique is not suitable for high-Pass or band –reject Filter

Q8:- What is the limitation of approximation method

A8:- This technique is not suitable for high-Pass or band –reject Filter

Q9:-What is bilinear Transformation

A9:- bilinear Transformation is one-to one mapping from s-domain to the z-domain

Q10:-What are the various methods to design IIR filters

A10:- The various methods are

1) Approximation of derivatives
2) Impulse invariant method
3) Bilinear Transformation

EXPERIMENT NO. 6

AIM : To Design Finite Impulse Response.

FIR filter:

lowpass 1500 Hz,

High pass,2200hz,

Bandpass 1750 Hz,

Band stops 790 Hz.

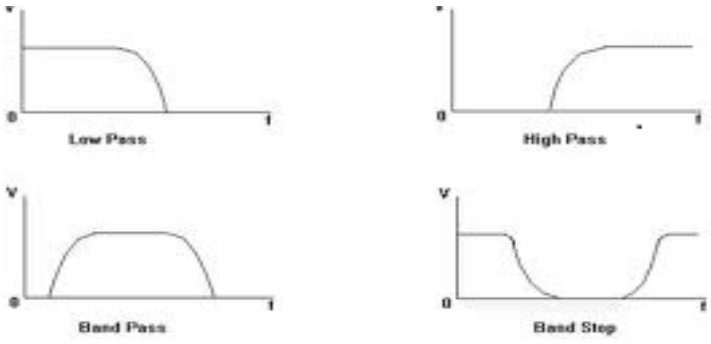
PROGRAM :

```
#include "DSK6713_AIC23.h"              //this file is added to initialize the DSK6713
#include "lowp1500.cof"                 // coefficient of low-pass filter file calculated from
                                           MATLAB
#include "highp2200.cof"                // coefficient of high-pass filter file calculated from
                                           MATLAB
#include "bpass1750.cof"                // coefficient of band-pass filter file calculated from
                                           MATLAB
#include "bstop790.cof"                 // coefficient of band-stop filter file calculated from
                                           MATLAB
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;      // set the sampling frequency, Different sampling
                                           Frequencies supported by AIC23 codec are 8,
                                           16, 24, 32, 44.1, 48, and 96 kHz.


short FIR_number = 0;                   //filter number
int yn = 0;                             //variable declaration

short dly[N];                           //declare delay buffer of n values
```

```
short h[4][N];                          //co-efficients of 4 different filters


interrupt void c_int11()                // ISR call, At each Interrupt, program execution goes
                                           to the interrupt service routine
{
        short i;                        //variable declaration
        dly[0] = input_sample();        //newest input @ top of buffer
        yn = 0;                         //initialize filter output
        for (i = 0; i< N; i++)          //for loop takes in the value of i from 0 to N
         yn +=(h[FIR_number][i]*dly[i]); //y(n) += h(LP#,i)*x(n-i)
         for (i = N-1; i > 0; i--)      //starting @ bottom of buffer
         dly[i] = dly[i-1];             //update delays with data move
          output_sample(yn >> 15);      //output filter,the value in the buffer yn indexed by
                                           the  variable loop is written on to the codec.
        return;                         // program execution goes back to while(1) and then
                                           again starts listening for next interrupt and this
                                           process goes on
}


void main()
{
        short i;                        //variable declaration
        for (i=0; i<N; i++)             //for loop which takes in the value of i from 0 to N=4
                                         and switches to corresponding filter co-efficients
```

```
    {
        dly[i] = 0;                 //init buffer
        h[0][i] = hlp[i];           //start addr of lp1500 coeff
        h[1][i] = hhp[i];           //start addr of hp2200 coeff
        h[2][i] = hbp[i];           //start addr of bp1750 coeff
        h[3][i] = hbs[i];           //start addr of bs790 coeff
    }
    comm_intr();                    // ISR function is called, using the given
                                       command
    while(1);                       //program execution halts and it starts listening
                                       for the interrupt which occur at every sampling
                                       period Ts.
}
```

OUTPUT :



QUESTION/ANSWER

Q1:-Give one advantage of digital filter over analog filter?
A1:- Digital filter can have characteristics which are not possible with analog filter
Q2:-What are main disadvantages of digital filter compared with analog filter?
A2:- Speed limitation
Q3:-What does IIR filter stand for
A3:- Infinite duration unit pulse response
Q4:-What does FIR filter stand for
A4:- Infinite duration unit pulse response
Q5:-what is the advantage of FIR filter over IIR filter
A5:- They are always stable, have exact linear phase
Q6:- What are the *methods* of designing FIR filters
A6:- **Parks-McClellan, Windowing, Direct Calculation**
A6:-Fourier series Method, Frequency Sampling Method
Q7:-What are the various window Techniques to design FIR filter
A7: Rectangular window, Hamming Window Function, Hanning window, Function, Blackman Window Function
Q8:-define the term Impulse Response *o*f FIR filter
A8- The "impulse response" of a FIR filter is actually just the set of FIR coefficients. (If you put an "impluse" into a FIR filter which consists of a "1" sample followed by many "0" samples, the output of the filter will be the set of coefficients, as the 1 sample moves past each coefficient in turn to form the output.)
Q9:- Define the term Transition Band
Transition Band - The band of frequencies between passband and stopband edges. The narrower the transition band, the more taps are required to implement the filter. (A "small" transition band results in a "sharp" filter.)
Q10:- What is the Z transform of a FIR filter?

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}$$

EXPERIMENT NO. 7

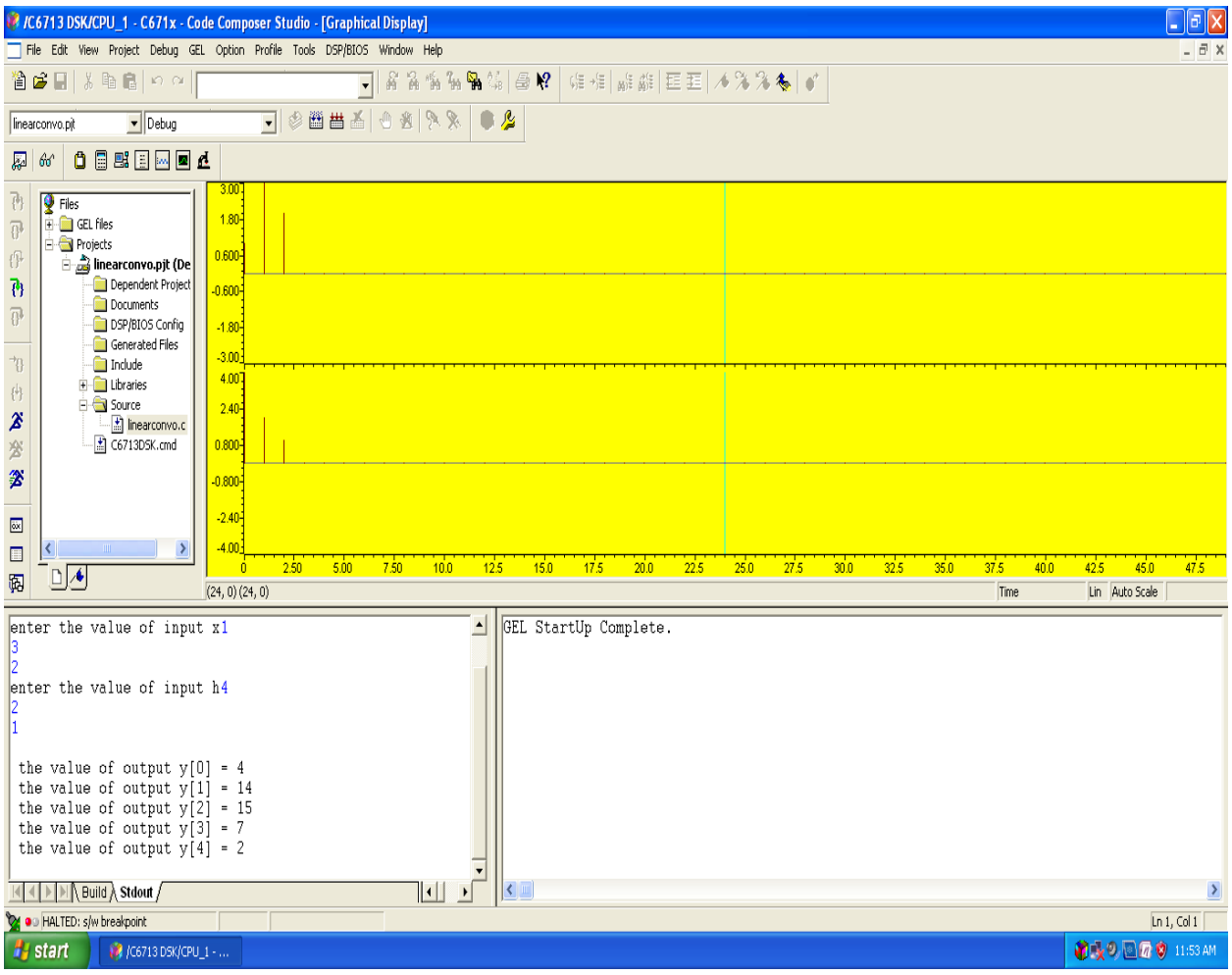
AIM : To study Linear Convolution technique.

PROGRAM :

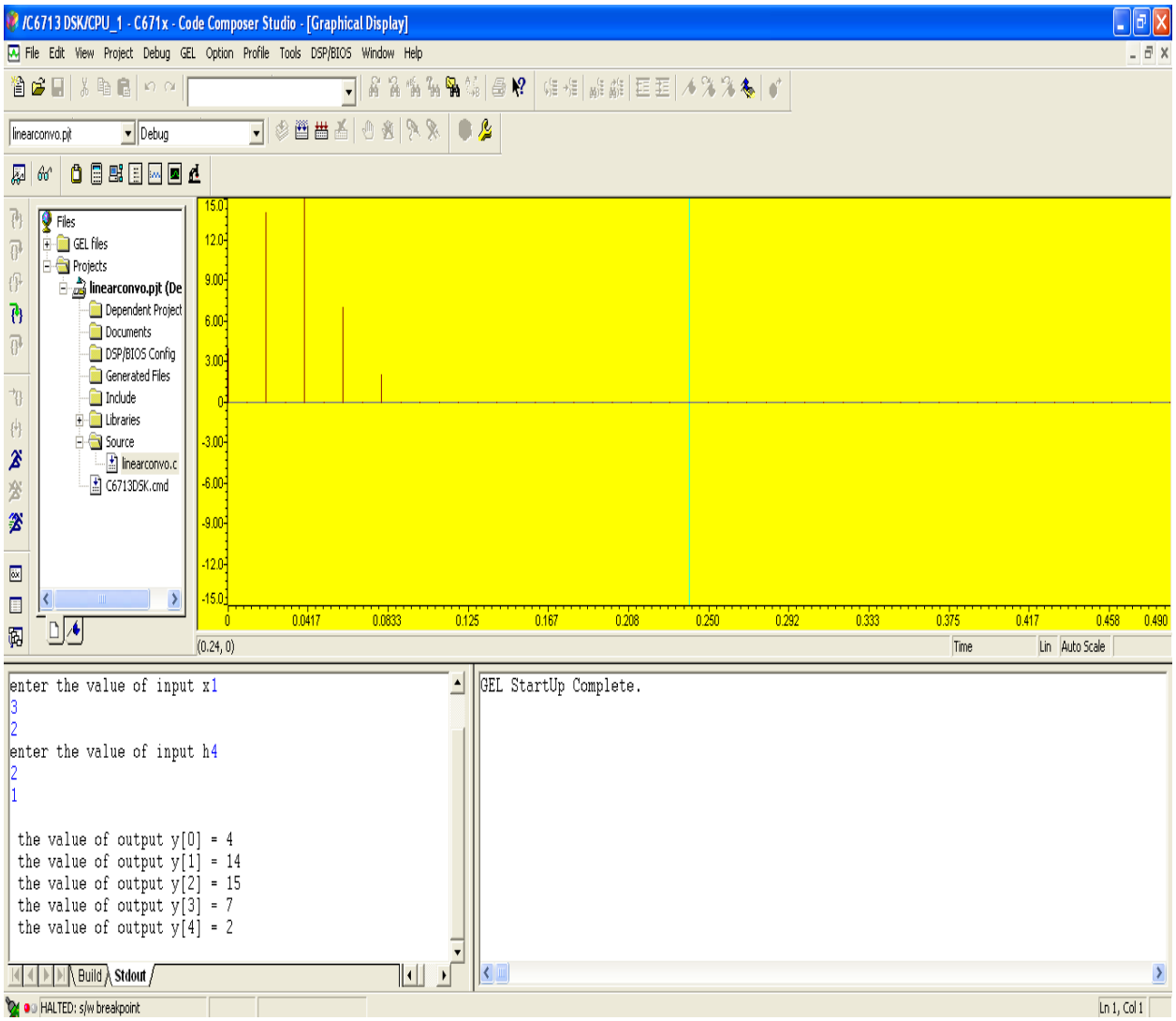
```
# include<stdio.h>                    // stdio.h, which stands for "standard input/output
                                        header", is the header in the C standard library that
                                        contains macro definitions, constants, and
                                        declarations of functions and types used for various
                                        standard input and output operations.

int x[15],h[15],y[15];                 // arrays initialization
void main()
{
        int i,j,m,n;                   // variable declaration

printf("enter the value of m");        // displays the line on the CCS screen & ask the user
                                          to enter the no. of values of first sequences.
scanf("%d",&m);                        // scans the user input & display it on the window
printf("enter the value of n");        // displays the line on the CCS screen & ask the user
                                          to enter  the no.  of values of second sequences
scanf("%d",&n);                        // scans the user input & display it on the window

printf("enter the value of input x");   // enter the input sequence. it takes the no. of digits
                                           as specified earlier
        for (i = 0;i<m;i++)            // in the loop it takes in the value of i less than m
        scanf("%d",& x[i] );           // scans the user input & display it on the window
printf("enter the value of input h");  // enter the input sequence. it takes the no. of digits as
                                          specified earlier
for (i = 0;i<n;i++)                    // in the loop it takes in the value of i less than n
        scanf("%d",& h[i] );           // scans the user input & display it on the window

                                       //padding of zeros
for (i = m;i<=m+n-1;i++)               // makes all values of x equlas to zero to avoid the
                                          garbage value
        x[i] = 0;
        for (i = n;i<=m+n-1;i++)       // makes all values of h equlas to zero
        h[i] = 0;

        // convolution operation
        for(i = 0;i<= m+n-1;i++)       // takes in values uptil m+n-1
        {
                y[i] = 0;
                for (j=0;j<=i;j++)
```

```
                    y[i] =y[i] + (x[j]* h[i-j]);        // second sequence i.e., h is shifted and is multiplied with the
                                                                first sequence x
        }
        for( i= 0;i<m+n-1;i++)
        printf("\n the value of output y[%d] = %d",i,y[i]);        // displays the line on the CCS window & displays the convolved output.

}
```

INPUT :



OUTPUT :


C6713 DSK/CPU_1 - C671x - Code Composer Studio - [Graphical Display]
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
linearconvo.pjt
Debug
Files
GEL files
Projects
linearconvo.pjt (De
Dependent Project
Documents
DSP/BIOS Config
Generated Files
Include
Libraries
Source
linearconvo.c
C6713DSK.cmd
15.0
12.0
9.00
6.00
3.00
0
-3.00
-6.00
-9.00
-12.0
-15.0
0
0.0417
0.0833
0.125
0.167
0.208
0.250
0.292
0.333
0.375
0.417
0.458
0.490
(0.24, 0)
Time
Lin
Auto Scale
enter the value of input x1
3
2
enter the value of input h4
2
1
the value of output y[0] = 4
the value of output y[1] = 14
the value of output y[2] = 15
the value of output y[3] = 7
the value of output y[4] = 2
Build
Stdout
GEL StartUp Complete.
HALTED: s/w breakpoint
Ln 1, Col 1

QUESTION/ANSWER

Q1:-how to classify signals?
discreet time/continous time, periodic/nonperiodic
Q2:-what is the use of random signals?
for testing systems dynamic response statistically for very small amplitudes and time durations.
Q3:-how to classify systems?
A3:-linear, stable, time invariant
Q4:-what's the difference between correlation & convolution? how to obtain convolution from correlation operator *?
A4:-conv(h,x) = h*(-x)
Q5:-what are the characteristics of a transient response of a system?
A5:-decay time, rise time, peak time, max overshoot, settling time
Q6.What is correlation?
A6 *Correlation* quantifies the strength of a linear relationship between two variables. When there is no correlation between two variables, then there is no tendency for the values of the variables to increase or decrease in tandem. Two variables that are uncorrelated are not necessarily independent, however, because they might have a nonlinear relationship.
Q7. What are the applications of correlation?
A7. It has applications that computes and plots the sample autocorrelation function (ACF) of a univariate, stochastic time series .
Q8. What is auto-correlation?
A8 The autocorrelation function of a random signal describes the general dependence of the values of the samples at one time on the values of the samples at another time.

Q9. What is cross-correlation?
A9 The cross correlation function however measures the dependence of the values of one signal on another signal.
Q10:- Give one disadvantages of Fourier ?
A10:-Fourier is not applicable for unit step, unit ramp and sinusoidal functions,this is overcome in Laplace.

EXPERIMENT NO. 8

AIM : To generate the Amplitude Shift Keying Modulation .

PROGRAM :

```
#include <std.h>                    //for input-output analyses
#include "rtdxaskcfg.h"             //this configuration file is added for Real time analysis.
                                      It is a BIOS file
#include "dsk6713_aic23.h"          //codec-DSK support file
#include "dsk6713.h"                //codec-DSK support file
#include <rtdx.h>                   //RTDX channel file is included
#include "target.h"                 //for init interrupt

#define BUFSIZE 64                   //# of points for buffer
#define BUFFERLENGTH 64
#define SINE_TABLE_SIZE              // Length of sine wave table
#define table_size 4                 // size of table=8
Uint32 fs = DSK6713_AIC23_FREQ_24KHZ; //set sampling rate

/* Codec configuration settings */
   DSK6713_AIC23_Config config =
{
   0x0017, // 0 DSK6713_AIC23_LEFTINVOL  Left line input channel volume
   0x0017, // 1 DSK6713_AIC23_RIGHTINVOL Right line input channel volume
   0x00d8, // 2 DSK6713_AIC23_LEFTHPVOL  Left channel headphone volume
   0x00d8, // 3 DSK6713_AIC23_RIGHTHPVOL Right channel headphone volume
   0x0011, // 4 DSK6713_AIC23_ANAPATH    Analog audio path control
   0x0000, // 5 DSK6713_AIC23_DIGPATH    Digital audio path control
   0x0000, // 6 DSK6713_AIC23_POWERDOWN  Power down control
   0x0043, // 7 DSK6713_AIC23_DIGIF      Digital audio interface format
   0x0001, // 8 DSK6713_AIC23_SAMPLERATE Sample rate control
   0x0001  // 9 DSK6713_AIC23_DIGACT     Digital interface activation
};
typedef Int sample;                          // representation of a data sample from A2D
sample inp_buffer[BUFSIZE];                   // Global declarations
sample out_buffer[BUFFERLENGTH];
Int volume=0;                                 // = MINVOLUME;  the scaling factor for
                                                 volume control
```

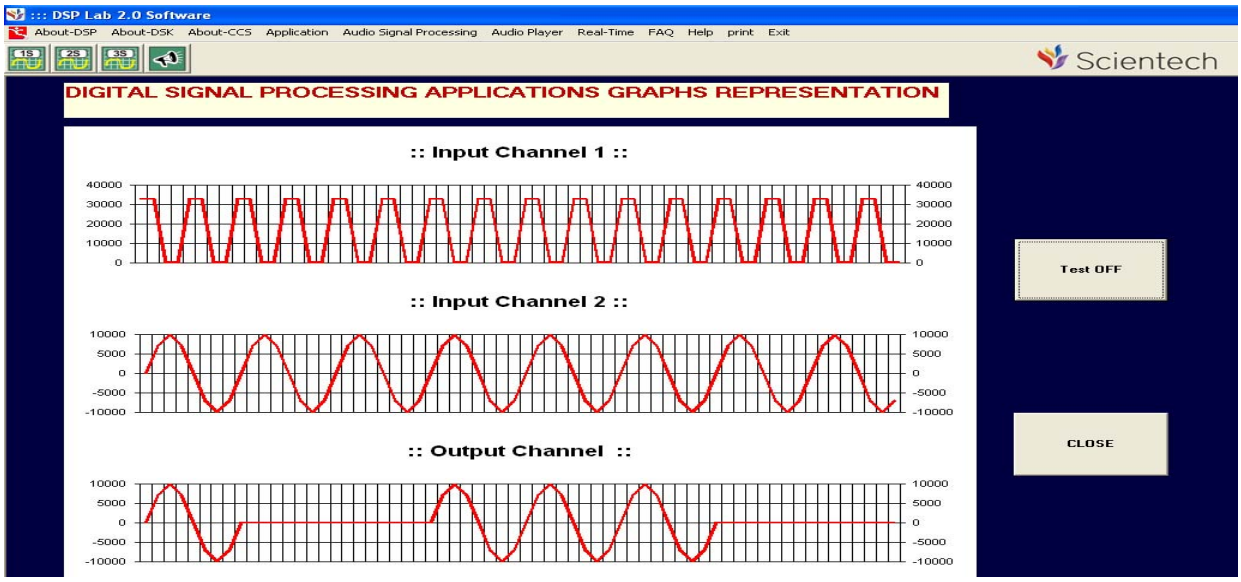
```
/* RTDX channels */
RTDX_CreateInputChannel(control_channel);                    // create contol channel
RTDX_CreateInputChannel(A2D_channel);                       // create input channel
RTDX_CreateOutputChannel(D2A1_channel);                     // create output channel1
RTDX_CreateOutputChannel(D2A2_channel);                     // create output channel2
RTDX_CreateOutputChannel(D2A3_channel);                      // create output channel3

int sine_table[SINE_TABLE_SIZE]={0,7070,10000,7070,0,-7070,-10000,-7070};
        //coefficients of Sine-wave
int out_buffer[64];                                          //output buffer
int i,j=0,k=0,msec=0;                                        //variable declaration
int l=0,m=0,loop=0;
int data_table[table_size];                                   //data table array
int sqr_data[64];                                            //data table array
int sine_buffer[64];                                          //data table array
sample *input = inp_buffer;                                   // inp_buffer data is stored into a
                                                                 variable of pointer type
Uns size = BUFSIZE;

void main()
{
       DSK6713_AIC23_CodecHandle hCodec;                      //initialize the CODEC
       DSK6713_init();                                        // Initialize the board support
                                                                 library, must be called  first
        hCodec = DSK6713_AIC23_openCodec(0, &config);     // Start the codec
        DSK6713_AIC23_setFreq(hCodec,fs);                     // Set the codec sample rate
frequency
         TARGET_INITIALIZE();                                 // Enable RTDX interrupt
         RTDX_enableInput(&control_channel);                  // enable volume control input
channel
         while (TRUE)
   {
          puts("Amplitude Shift Keying Example Started");       / /prints the line on CCS window
          if (!RTDX_channelBusy(&control_channel))             // Read a new volume when the
                                                                  hosts  send it
      {
          RTDX_readNB(&control_channel, &volume, sizeof(volume));
       }
           while (!RTDX_isInputEnabled(&A2D_channel))         // checks if Input channel (A2D)
                                                                  of  RTDX channel is not
```

```
                                                                    enabled
    {
      #if     RTDX_POLLING_IMPLEMENTATION
      RTDX_Poll();        /* poll comm channel for input*/
      #endif
     }
        /*
          *  A2D: get digitized input (get signal from the host through RTDX).
         *  If A2D_channel is enabled, read data from the host.
           */
        RTDX_read(&A2D_channel, input, size*sizeof(sample));     // read data by DSK
        /* fill the data table to genarate the square table*/
          for(i=0; i<=table_size/2; i++)                          //set 1st half of buffer
         data_table[i] = 0x7FFF;                                  //with max value (2^15)-1
                  for(i=table_size/2;i<table_size;i++)           //set 2nd half of buffer
                    data_table[i] = 0;                            //with -(2^15)
                     i=0;
                              /* store the squarewave in sqr_data to display on the graph*/
                              for(i=0; i<table_size/2; i++)
                              {
                                       sqr_data[k] = data_table[i];            //output to buffer
                                       k++;
                                       if(k==BUFFERLENGTH) k=0;
                              }
                                       for(i=table_size/2;i<table_size;i++)
                                       {
                                              sqr_data[k] = data_table[i];       //output to buffer
                                               k++;
                                               if(k==BUFFERLENGTH) k=0;
                                       }
RTDX_write(&D2A1_channel,sqr_data, size*sizeof(sample));                 // sends the sqr_data
                                                                            value  on
                                                                           outputchannel1

(D2A1) of the RTDX

while(RTDX_writing)
                                       {
                                          #if     RTDX_POLLING_IMPLEMENTATION
                                            RTDX_Poll();        /* poll comm channel for output */
```

```
                                    #endif
                                   }
                                      for(loop=0;loop<SINE_TABLE_SIZE;loop++)
                                {
                                      sine_buffer[j]= sine_table[loop];
                                       j++;
                                       if (j== BUFFERLENGTH)
                                       j=0;
                                }
RTDX_write(&D2A2_channel,sine_buffer, size*sizeof(sample));       // sends the sine_buffer
                                                                     value on  output
                                                                     channel1 (D2A1) of the
                                                                  RTDX while(RTDX_writing)
                                {
                                          #if
RTDX_POLLING_IMPLEMENTATION
                                    RTDX_Poll();       /* poll comm channel for output */
                                        #endif
                                     }

                                     /* store the result in to out_buffer*/
                                        for(i=0;i<table_size;i++)
                                   {
                                      if(data_table[i]>0)
                                   {

      for(loop=0;loop<SINE_TABLE_SIZE;loop++)
                                {
                                      out_buffer[j] = sine_table[loop];     //output to buffer
                                       j++;                                 //increment buffer count
                                  if(j==BUFFERLENGTH) j=0;                  //if @ bottom reinit
                                                                              count

                                    }
                                     }
                                       else
for(loop=0;loop<SINE_TABLE_SIZE;loop++)
                                    {
                                       out_buffer[j] = 0;         //output to buffer
                                       j++;                       //increment buffer count
```

```
if(j==BUFFERLENGTH) j=0;            //if @ bottom reinit count

                                                    }
                                                }
RTDX_write(&D2A3_channel,out_buffer, size*sizeof(sample));      // sends the out_buffer value
                                                                    on output  channel1(D2A1)
                                                                    of the RTDX
                                                                    while(RTDX_writing)




                                                {
                                                        #if
RTDX_POLLING_IMPLEMENTATION
                                                          RTDX_Poll();       /* poll comm channel for
                                                                                output */

                                                        #endif
                                                }
        }                           //end of while-loop
          }                          // end of main program
```

OUTPUT :



QUESTION/ANSWER :-

Q1:- ASK ,PSK ,FSK and QAM are examples of which conversion
A1:- Analog to Digital Conversion
Q2:-AM and FM are the examples of which conversion
A2:-Analog To Analog Conversion
Q3:-IF baud rate is 400 for QPSK ,what is the bit rate
A3:-800 bps
Q4:-How many carrier frequencies are used in BASK
A4:-one
Q5:-What is modulation
A5:- The technique of superimposing the message signal on the carrier is known as modulation
Q6:-What is the purpose of digital modulation
A6:-The purpose of digital modulation is to convert an information bearing discrete-time symbol into a continuous-time waveform
Q7:-What does BASK stand for
A7:- Binary Amplitude shift key
Q8:-define Modulation
A8:- modulation is the process of varying one or more properties of a high-frequency periodic waveform, called the carrier signal, with a modulating signal which typically contains information to be transmitted
Q9:- Define Pulse modulation
A9:- transfers a narrowband analog signal over an analog baseband channel as a two-level signal by modulating a pulse wave.
Q10:- Define Amplitude modulation
A10:- It is a technique in which amplitude of the carrier signal is varied in accordance to the instantaneous amplitude of the modulating signal

EXPERIMENT NO. 9

AIM : To generate the Frequency shift keying Modulation .

PROGRAM :

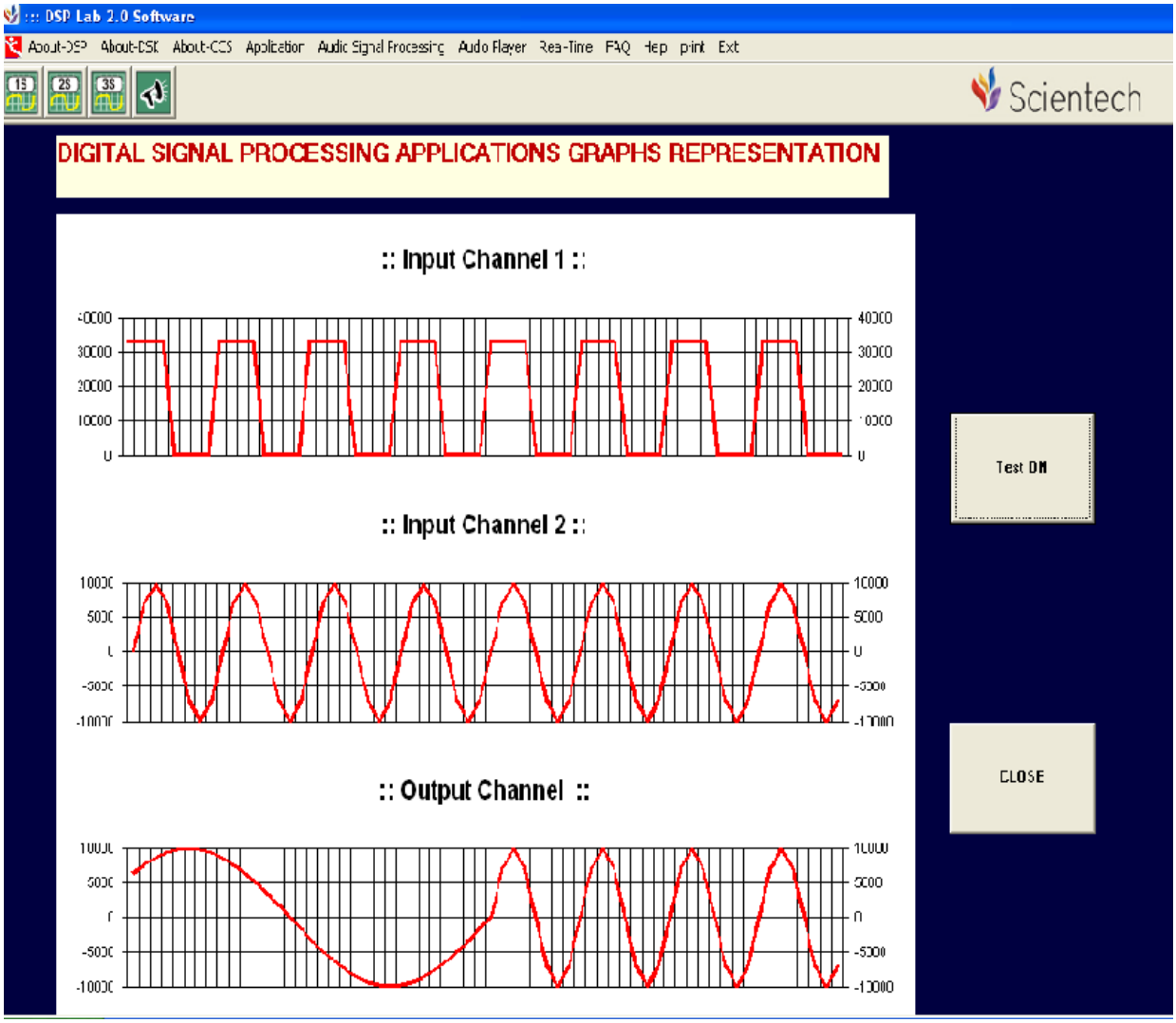
```
#include<std.h>                          //for input-output analyses
#include "rtdxfskcfg.h"                  //this configuration file is added for Real time
                                           analysis. It is a BIOS file

#include "dsk6713_aic23.h"               //codec-DSK support file
#include "dsk6713.h"                     //codec-DSK support file
#include <rtdx.h>                        //RTDX channel file is included
#include "target.h"                      //for init interrupt
#define BUFSIZE 64                       //# of points for buffer
#define BUFFERLENGTH 256
/* Length of sine wave table */
#define SINE_TABLE_SIZE 8
#define SINE_TABLE 36
/* size of table=8*/
#define table_size 8
/* Codec configuration settings */
DSK6713_AIC23_Config config = {
   0x0017,                               // 0 DSK6713_AIC23_LEFTINVOL  Left line input
                                            channel volume
   0x0017,                               // 1 DSK6713_AIC23_RIGHTINVOL Right line
                                            input channel volume

   0x00d8,                               // 2 DSK6713_AIC23_LEFTHPVOL  Left channel
                                            headphone volume
   0x00d8,                               // 3 DSK6713_AIC23_RIGHTHPVOL Right
                                            channel headphone volume

   0x0011,                               // 4 DSK6713_AIC23_ANAPATH    Analog audio
                                            path control
   0x0000,                               // 5 DSK6713_AIC23_DIGPATH    Digital audio
                                            path control
   0x0000,                               // 6 DSK6713_AIC23_POWERDOWN  Power
                                           down control
   0x0043,                               // 7 DSK6713_AIC23_DIGIF Digital audio
                                            interface format
   0x0001,                               // 8 DSK6713_AIC23_SAMPLERATE Sample
                                            rate control
   0x0001                                // 9 DSK6713_AIC23_DIGACT     Digital
                                            interface activation
};
```

```
typedef Int sample;                          // representation of a data sample from A2D
/* Global declarations */
sample inp_buffer[BUFSIZE];
sample out_buffer[BUFFERLENGTH];
Int volume=0;                                // = MINVOLUME; the scaling factor for volume
                                                control
/* RTDX channels */
RTDX_CreateInputChannel(control_channel);    // create contol channel
RTDX_CreateInputChannel(A2D_channel);        // create input channel
RTDX_CreateOutputChannel(D2A1_channel);      // create output channel1
RTDX_Create Output Channel(D2A2_channel);    // create output channel2
RTDX_Create Output Channel(D2A3_channel);    // create output channel3
Uint32 fs = DSK6713_AIC23_FREQ_24KHZ;        //set sampling rate
int sine_table[SINE_TABLE_SIZE]={0,7070,10000,7070,0,-7070,-10000,-7070};
//coefficients of

Sine- wave
int sin_table[SINE_TABLE] =
 {                                           //coefficients of square-wave
    0,   1736, 3420, 5000, 6427, 7660, 8660,  9396, 9848,
    10000, 9848, 9396, 8660, 7660, 6427, 5000, 3420,  1736,
     0,   -1736, -3420,-5000,-6427,-7660,-8660,-9396,-9848,
    -10000, -9848,-9396,    -8660,-7660,-6427,-5000,-3420,-1736};

int out_buffer[BUFFERLENGTH];                //output buffer
int i;                                       //for buffer count
int j=0,k=0,msec=0;                          //variable declaration
int l=0,m=0,loop=0;
int data_table[table_size];                  //data table array
int sqr_data[BUFFERLENGTH];                  //data table array
int sine_buffer[BUFFERLENGTH];               //data table array
sample *input = inp_buffer;                  // inp_buffer data is
                                                stored  into a
                                                variable of pointer
                                                type


Uns size = BUFSIZE;
void main()
{
        TARGET_INITIALIZE();                 // Enable RTDX interrupt
          RTDX_enableInput(&control_channel); //enable volume control
                                                input channel
          while (TRUE)
    {
```

```
        puts("Frequency  Shift Keying Example Started");          //prints the line on CCS
                                                                          window
        if (!RTDX_channelBusy(&control_channel))                 // Read a new volume when
                                                                    the hosts send it
    {
       RTDX_readNB(&control_channel, &volume, sizeof(volume));
     }
              while (!RTDX_isInputEnabled(&A2D_channel))
       {
              #if    RTDX_POLLING_IMPLEMENTATION
                     RTDX_Poll();                          /* poll comm channel for input*/
              #endif
     }
          RTDX_read(&A2D_channel, input, size*sizeof(sample));
         /* fill the data table to genarate the square table*/
for(i=0; i<=table_size/2; i++)                                    //set 1st half of buffer
data_table[i] = 0x7FFF;                                           //with max value (2^15)-1
for(i=table_size/2;i<table_size;i++)                              //set 2nd half of buffer
data_table[i] = 0; //with -(2^15)
i=0;
                                   /* store the squarewave in sqr_data to display on the
graph*/
for(i=0; i<table_size/2; i++)
{
sqr_data[k] = data_table[i];                                      //output to buffer
k++;
if(k==BUFFERLENGTH) k=0;
}
for(i=table_size/2;i<table_size;i++)
{
sqr_data[k] = data_table[i];                                      //output to buffer
k++;
if(k==BUFFERLENGTH) k=0;
}
//i=0;
RTDX_write(&D2A1_channel,sqr_data, size*sizeof(sample));
         //printf("hello");
         while(RTDX_writing)
{
#if    RTDX_POLLING_IMPLEMENTATION
       RTDX_Poll();                                  /* poll comm channel for output */
#endif
      }
for(m=0;m<SINE_TABLE_SIZE;m++)
{
```

```
sine_buffer[l] = sine_table[m];                                    //output to buffer
l++;
if(l==BUFFERLENGTH) l=0;                                           //if @ bottom reinit count
}
RTDX_write(&D2A2_channel,sine_buffer, size*sizeof(sample));
          //printf("hello");
          while(RTDX_writing){
#if     RTDX_POLLING_IMPLEMENTATION
          RTDX_Poll();                                             /* poll comm channel for output */
#endif
       }
                                                                   /* store the result in to out_buffer*/
for(i=0;i<table_size;i++)
{
if(data_table[i]>0)
{
for(loop=0;loop<SINE_TABLE_SIZE;loop++)
{
out_buffer[j] = sine_table[loop];                                  //output to buffer
j++;                                                               //increment buffer count
if(j==BUFFERLENGTH) j=0;                                           //if @ bottom reinit count
}
}
else for(loop=0;loop<SINE_TABLE;loop++)
{
out_buffer[j] = sin_table[loop];                                   //output to buffer
j++;                                                               //increment buffer count
if(j==BUFFERLENGTH) j=0;                                          //if @ bottom reinit count
}
}
//loop=0;
//i=0;
                    RTDX_write(&D2A3_channel,out_buffer, size*sizeof(sample));
                    //printf("hello");
          while(RTDX_writing)
{
#if     RTDX_POLLING_IMPLEMENTATION
         RTDX_Poll();                                              /* poll comm channel for output */
#endif
       }
}
}
```

OUTPUT :


::: DSP Lab 2.0 Software
About-DSP
About-DSK
About-CCS
Application
Audio Signal Processing
Audio Player
Real-Time
FAQ
Help
print
Exit
Scientech
DIGITAL SIGNAL PROCESSING APPLICATIONS GRAPHS REPRESENTATION
:: Input Channel 1 ::
:: Input Channel 2 ::
:: Output Channel ::
Test DM
CLOSE

QUESTION /ANSWERS:

Q1.What is the use of the command “include <std.h>”

A1.for input-output analyses

Q2. What is the use of the command “include rtdxfskcfg.h “

A2.In this command the configuration file is added for Real time
Q3.What is the use of for loop
A3. The for loop is used mainly to increment variables, or count or sort a quantity of data (usually stored in an array).
Q4.Why do we use the command STRLEN in c
A4. strlen calculates the number of characters in the given string

Q5 Why do we use of the command “STRREV” in C language

A5.”strrev reverses all characters in the given string (except the terminating character) and returns a pointer to the reversed string

Q6.Difference between deterministic and non deterministic signal

A6. Deterministic signal are random in nature

Q7.What is the difference between Fourier series and transform

A7.Fourier series is drawn for periodic signal and Fourier transform for non periodic signal

Q8.What is meant by discrete time processing

A8.discrete time processing is done by transforming the continous time signal into small interval by sampling the signal into discrete intervals

Q9 What is the purpose of digital modulation
A9:-The purpose of digital modulation is to convert an information bearing discrete-time symbol into a continuous-time waveform
Q10.What is digital filter
A10. Digital filter is one in which both the i/p and o/p are discrete time signal

EXPERIMENT NO.10

AIM: -To Generate the sine ,square, triangular wave using lookup table and produces an output stream.

PROGRAMS: -

Sine Wave: -

**Program To generate the sine wave.

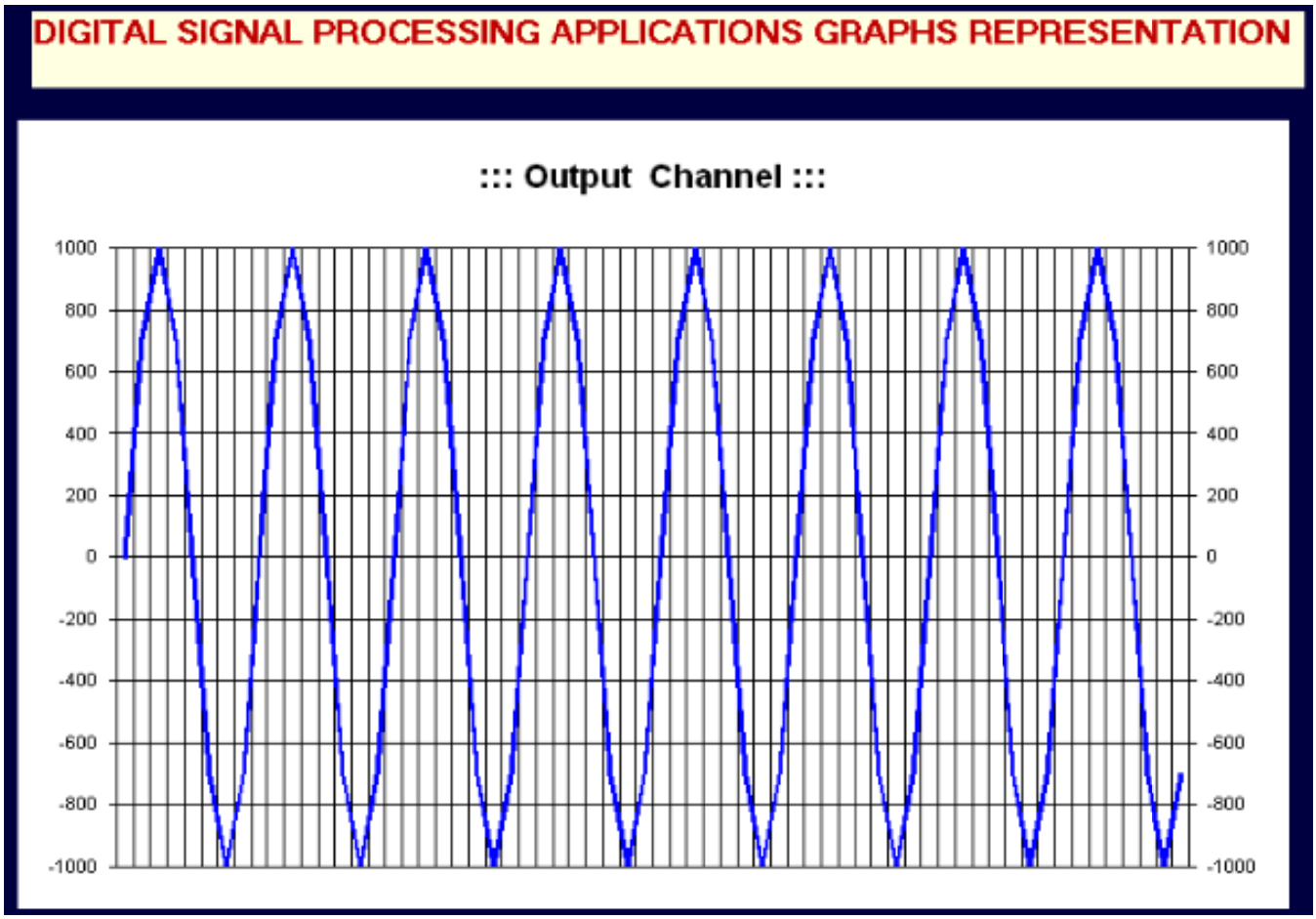
```
#include <std.h>
#include <log.h>
#include <rtdx.h>                          //for rtdx support
#include "rtdxsinecfg.h"                   //this configuration file is added for Real time
                                           analysis. It is a BIOS file
#include "target.h"                        //for init interrupt

#define BUFSIZE 64
#define BUFFERLENGTH 64
#define MINVOLUME 1

typedef Int sample;                                  // representation of a data sample
from A2D
sample inp_buffer[BUFSIZE];                // Global declarations
int out_buffer[BUFFERLENGTH];
Int volume=0;                                        // = MINVOLUME; the scaling factor for
```

```
                                                    Volume control
/* RTDX channels */
RTDX_CreateInputChannel(control_channel);              // create contol channel
RTDX_CreateInputChannel(A2D_channel);                  // create input channel
RTDX_CreateOutputChannel(D2A1_channel);                // create output channel
/*
 * ======== main ========
*/
Void main()
{
   sample *input = inp_buffer;                         // inp_buffer data is stored into a variable of
                                                       pointer type.
   Uns size = BUFSIZE;
   int sin_table[8] = {0,707,1000,707,0,-707,-1000,-707};      //look-up table for sine-wave
generation
   int i=0,loop=0;                                     // variable declaration

   TARGET_INITIALIZE();                                // init for interrupt

   LOG_printf(&trace,"\n Sine Wave Example Started");       //print the specified line on CCS
                                                       window
   RTDX_enableInput(&control_channel);                 // enable input channel for control
        while (TRUE)                                   // Infinite loop
        {
```

```
            if (!RTDX_channelBusy(&control_channel)     // if control channel not busy
    {
      RTDX_readNB(&control_channel, &volume, sizeof(volume));      // read from PC
    }
while (!RTDX_isInputEnabled(&A2D_channel))              // checks if Input channel (A2D) of RTDX
                                                        channel is not enabled
      {
            #if     RTDX_POLLING_IMPLEMENTATION         //polling uses a continuous
                                                          procedure of testig when
                                                          the data is ready

            RTDX_Poll();                                // poll comm channel for input
            #endif
    }

    /*
     *  A2D: get digitized input (get signal from the host through RTDX).
     *  If A2D_channel is enabled, read data from the host.
     */
        RTDX_read(&A2D_channel, input, size*sizeof(sample));    // read data by DSK

    /*
     *  D2A: produce analog output (send signal to the host through RTDX).
     *  If D2A_channel is enabled, write data to the host.
```

```
 */
            out_buffer[i]= sin_table[loop];          // puts the sine_table points &
                                                     stores it into output buffer
            i++;                                     // increment by 1 data from PC
            if (i== BUFFERLENGTH)                // takes in the value of sine-
                                                     wave uptil the bufferlength
            i=0;
            if (++loop >7)
            loop = 0;
RTDX_write(&D2A1_channel,out_buffer, size*sizeof(sample)   // send data from DSK to
                                                                RTDX channel
            printf("hello");                                    // prints "hello" on the
                                                                CCS window
    while(RTDX_writing)                                    // for writing on RTDX
                                                                channel
{
    #if    RTDX_POLLING_IMPLEMENTATION
     RTDX_Poll();                                       // poll comm channel for output
   #endif
}  }}
```

Output: -


DIGITAL SIGNAL PROCESSING APPLICATIONS GRAPHS REPRESENTATION
::: Output Channel :::
1000
800
600
400
200
0
-200
-400
-600
-800
-1000

Square Wave: -

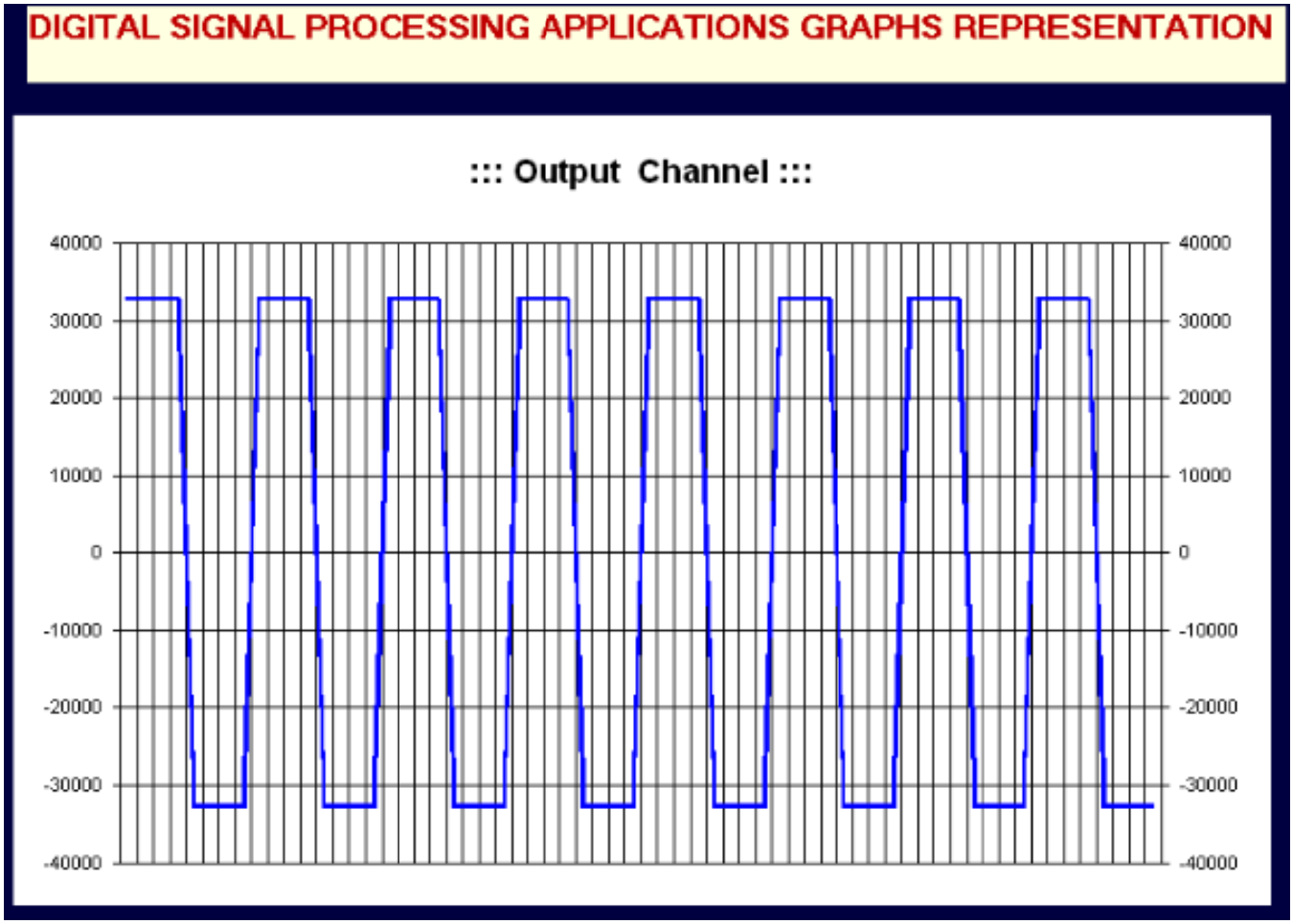
**Program To generate the Square wave.

```
#include <std.h>
#include "rtdxsqurecfg.h"      //this configuration file is added for Real time analysis. It is a BIOS
file
#include "dsk6713_aic23.h"          //this file is added to initialize the DSK6713
#include "dsk6713.h"
#include <log.h>
#include <rtdx.h>                          //for rtdx support
#include "target.h"                        //for init interrupt

#define BUFSIZE 64                         // define the buffersize
#define BUFFERLENGTH 64                    // define the bufferlength
#define table_size 8                  //size of table=48

typedef Int sample;                   // representation of a data sample from A2D
sample inp_buffer[BUFSIZE];           // Global declarations
sample out_buffer[BUFFERLENGTH];
Int volume=0;                         // = MINVOLUME; the scaling factor for volume control
/* RTDX channels */
```

```
RTDX_CreateInputChannel(control_channel);          // create contol channel
RTDX_CreateInputChannel(A2D_channel);              // create input channel
RTDX_CreateOutputChannel(D2A1_channel);                    // create output channel
int data_table[table_size];     //data table array
sample *input = inp_buffer;              // inp_buffer data is stored into a variable of pointer type.
Uns size = BUFSIZE;
int i=0,j=0;                                       // variable daclaration
/*
* ======== main ========
*/
Void main()
{
        TARGET_INITIALIZE();                       // Enable RTDX interrupt
LOG_printf(&trace,"\n Square Wave Example Started");     //print the specified line on CCS
window
        RTDX_enableInput(&control_channel);        // enable volume control input channel
        while (TRUE)
        {
                if (!RTDX_channelBusy(&control_channel))  // Read a new volume when the
hosts send it
                {
                        RTDX_readNB(&control_channel, &volume, sizeof(volume));         // read
from PC
                }
        while (!RTDX_isInputEnabled(&A2D_channel))
                {
                        #if    RTDX_POLLING_IMPLEMENTATION
                RTDX_Poll();       // poll comm channel for input
                        #endif
                }
     /*
      *  A2D: get digitized input (get signal from the host through RTDX).
      *  If A2D_channel is enabled, read data from the host.
      */
        RTDX_read(&A2D_channel, input, size*sizeof(sample));      // read data by DSK
     /*
      *  D2A: produce analog output (send signal to the host through RTDX).
```

```
*  If D2A_channel is enabled, write data to the host.
*/
            for(i=0; i<=table_size/2; i++) //set 1st half of buffer
            {
                  data_table[i] = 0x7FFF; //with max value (2^15)-1
            }
            for(i=table_size/2;i<table_size;i++) //set 2nd half of buffer
            {
                  data_table[i] = -0x8000; //with -(2^15)
            }
            i = 0;
            for(i=0; i<table_size/2; i++)
            {
                  out_buffer[j] = data_table[i]; //output to buffer
                  j++;
                  if(j==BUFFERLENGTH) j=0;
            }
            for(i=table_size/2;i<table_size;i++)
            {
                  out_buffer[j] = data_table[i]; //output to buffer
                  j++;
                  if(j==BUFFERLENGTH) j=0;
            }
            i=0;
RTDX_write(&D2A1_channel,out_buffer, size*sizeof(sample));
            printf("hello");          // prints "hello" on the CCS window
            while(RTDX_writing)             // for writing on RTDX channel
            {
                  #if    RTDX_POLLING_IMPLEMENTATION
                 RTDX_Poll();      // poll comm channel for output
                  #endi
          }
           }
          }
```

Output: -


DIGITAL SIGNAL PROCESSING APPLICATIONS GRAPHS REPRESENTATION
::: Output Channel :::
40000
30000
20000
10000
0
-10000
-20000
-30000
-40000

Triangular Wave: -

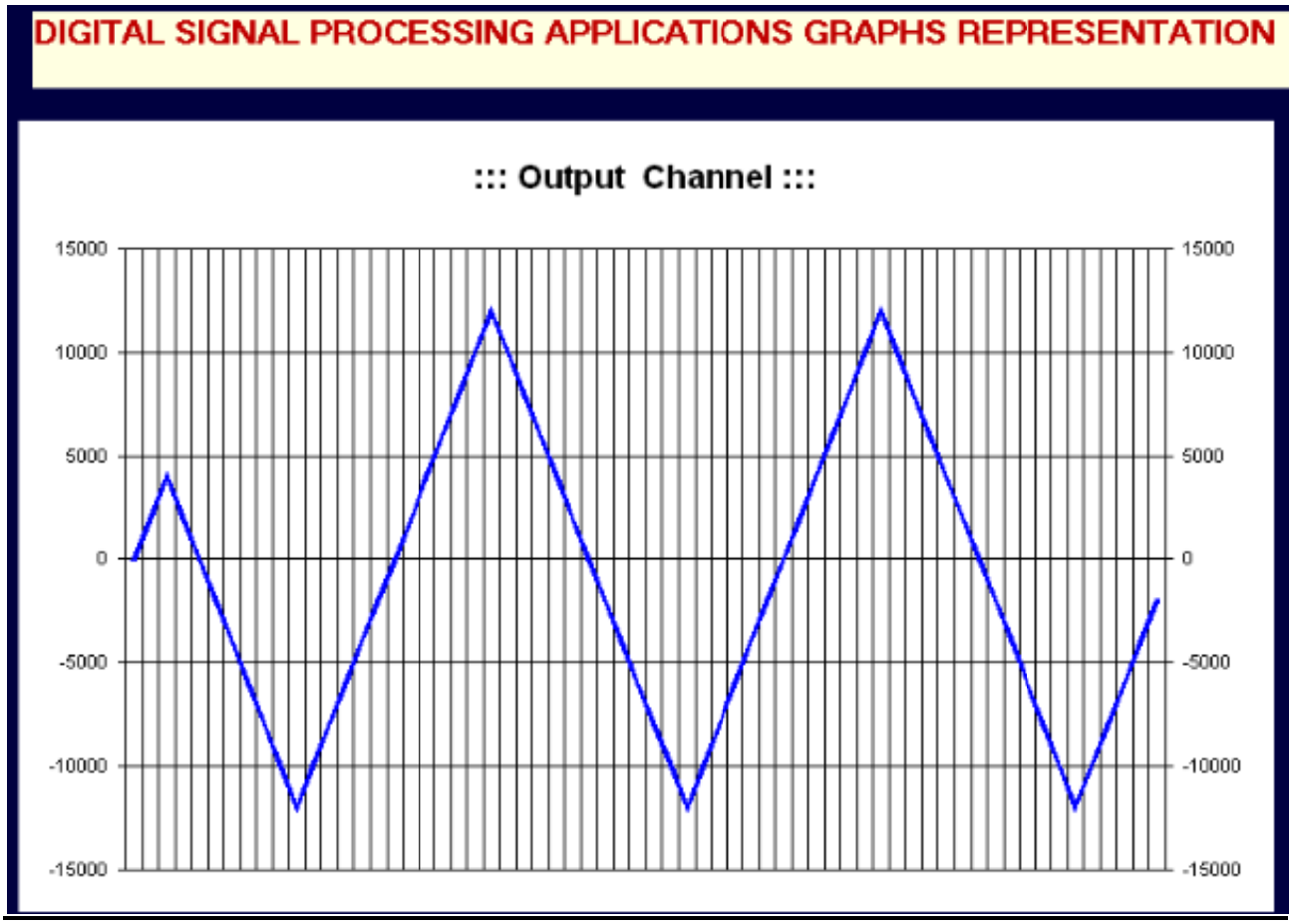
**Program To generate the Triangular wave.

```
#include <std.h>
 #include"rtdxtrangcfg.h"       //this configuration file is added for Real time analysis.
 #include "dsk6713_aic23.h"            //this file is added to initialize the DSK6713
 #include "dsk6713.h"
 #include <log.h>
 #include <rtdx.h>                              //for rtdx support
 #include "target.h"                            //for init interrupt
 #define BUFSIZE 64                             // define the buffersize
 #define BUFFERLENGTH 64                        // define the bufferlength
 #define TABLE_SIZE   24                        // Length of sine wave table
 typedef Int sample;                    // representation of a data sample from A2D
 sample inp_buffer[BUFSIZE];            // Global declarations
 Int volume=0;                  // = MINVOLUME; the scaling factor for volume control
 /* RTDX channels */
 RTDX_CreateInputChannel(control_channel);          // create control channel
 RTDX_CreateInputChannel(A2D_channel);              // create input channel
 RTDX_CreateOutputChannel(D2A1_channel);                    // create output channel
 int out_buffer[64];                            //output buffer
 int loop=0,i=0;                                        // variable daclaration
 sample *input = inp_buffer;   // inp_buffer data is stored into a variable of pointer type.
 Uns size = BUFSIZE;
 int trang_table[TABLE_SIZE]=           // co-efficients (points) for trangular wave generation
 {
  0,2000,4000,6000,8000,10000,12000,10000,8000,6000,
  4000,2000,0,-2000,-4000,-6000,-8000,-10000,-12000,-10000,
  -8000,-6000,-4000,-2000
 };                 //table values
void main()
```

```
    {
        TARGET_INITIALIZE();       // Enable RTDX interrupt
       LOG_printf(&trace,"\n Trangularwave  example started");    //print the specified line on CCS
      RTDX_enableInput(&control_channel);          // enable volume control input channel
       while (TRUE)
 {
        if (!RTDX_channelBusy(&control_channel)) // Read a new volume when the h

 {
      RTDX_readNB(&control_channel, &volume, sizeof(volume));          // read from PC

 }
   while (!RTDX_isInputEnabled(&A2D_channel))  // checks if Input channel (A2D) of RTDX channel
                                                 is not enabled
   {
                        #if    RTDX_POLLING_IMPLEMENTATION
               RTDX_Poll();       // poll comm channel for input
                        #endif
   }
   /*
    *  A2D: get digitized input (get signal from the host through RTDX).
    *  If A2D_channel is enabled, read data from the host.
    */
      RTDX_read(&A2D_channel, input, size*sizeof(sample));      // read data by DSK
   /*
    *  D2A: produce analog output (send signal to the host through RTDX).
    *  If D2A_channel is enabled, write data to the host.
    */
               out_buffer[i] = trang_table[loop]; //output to buffer
               i++;
               if(i==BUFFERLENGTH) i=0; //if @ bottom reinit count
               if (++loop > 23)
               loop = 0;
RTDX_write(&D2A1_channel,out_buffer, size*sizeof(sample));       // send data from DSK to RTDX c

          printf("hello");                    // prints "hello" on the CCS window
         while(RTDX_writing)                        // for writing on RTDX channel
```

```
{
        #if    RTDX_POLLING_IMPLEMENTATION
       RTDX_Poll();       // poll comm channel for output
        #endif
}
}
}
```

Output: -



QUESTION/ANSWER

Q1.What is look up table

A1. A LookupTable is an unordered collection of values; each value indexed by a "key," which is a value of any type that's used to look up a value stored in the collection. In effect, this class provides what some programming languages call an "associative array," because it allows a value to be associated with an arbitrary key, and then efficiently found given the same key.

Q2. Which operator we will use to create look up table

A2: local tab = new LookupTable();

Q3 . What is a local block?

A local block is any portion of a C program that is enclosed by the left brace ({) and the right brace (}). A C function contains left and right braces, and therefore anything between the two braces is contained in a local block. An if statement or a switch statement can also contain braces, so the portion of code between these two braces would be considered a local block.

4. When is a switch statement better than multiple if statements?

A switch statement is generally best to use when you have more than two conditional expressions based on a single variable of numeric type.

5. Is a default case necessary in a switch statement?

No, but it is not a bad idea to put default statements in switch statements for error- or logic-checking purposes.

6. Can the last case of a switch statement skip including the break?

Even though the last case of a switch statement does not require a break statement at the end, you should add break statements to all cases of the switch statement, including the last case. You should do so primarily because your program has a strong chance of being maintained by someone other than you who might add cases but neglect to notice that the last case has no break statement.

This oversight would cause what would formerly be the last case statement to "fall through" to the new statements added to the bottom of the switch statement. Putting a break after each case statement would prevent this possible mishap and make your program more "bulletproof." Besides, most of today's optimizing compilers will optimize out the last break, so there will be no performance degradation if you add it.

7. Other than in a for statement, when is the comma operator used?

The comma operator is commonly used to separate variable declarations, function arguments, and expressions, as well as the elements of a for statement.

8. How can you tell whether a loop ended prematurely?

Generally, loops are dependent on one or more variables. Your program can check those variables outside the loop to ensure that the loop executed properly.

9. What is the difference between goto and long jmp( ) and setjmp()?

A goto statement implements a local jump of program execution, and the longjmp() and setjmp() functions implement a nonlocal, or far, jump of program execution. Generally, a jump in execution of any kind should be avoided because it is not considered good programming practice to use such statements as goto and longjmp in your program.

A goto statement simply bypasses code in your program and jumps to a predefined position. To use the goto statement, you give it a labeled position to jump to. This predefined position must be within the same function.

10. 9. What is an l value?

An l value is an expression to which a value can be assigned. The lvalue expression is located on the left side of an assignment statement, whereas an rvalue is located on the right side of an assignment statement. Each assignment statement must have an lvalue and an rvalue. The lvalue expression must reference a storable variable in memory. It cannot be a constant.