DSPA:

1.Hello Lcdk:

```
#include<stdio.h>
void main(){
printf(“HELLO LCDK”);
}
```

2.SINE WAVE GENERATION:

```
#include<stdio.h>
#include<math.h>
#define FREQ 500
float m[128];
main()
{
    int i = 0;
    for(i=0;i<127;i++)
    {
        m[i] = sin(2*3.14*FREQ*i/24000);
        printf("%f\n",m[i]);
    }
}
```

RANDOM WAVE GENERATION:

```
#include<stdio.h>
#include<math.h>
float m[128];
main()
 {
    int i=0;
    for(i=0;i<127;i++)
    {
        m[i] = i;
        printf("%f \n",m[i]);
    }
 }
```

TRIANGULAR WAVE GENERATION:

```
#include<stdio.h>
#include<math.h>
#define FREQ 500
float m[101];
main()
{
    int i = 0,j=0;
    int t=0;
    for(i=0;i<50;i++)
    {
        m[i] = t++;
        printf("%f \n", m[i]);
    }
    for(j=50;j<100;j++)
    {
        m[j] = t--;
        printf("%f\n",m[j]);
    }
}
```

3.LINEAR CONVOLUTION:

```
#include<stdio.h>
#define LENGHT1 6 /*Lenght of i/p samples sequence*/
#define LENGHT2 4 /*Lenght of impulse response Co-efficients */
int x[2*LENGHT1-1]={1,2,3,4,5,6,0,0,0,0,0}; /*Input Signal Samples*/
int h[2*LENGHT1-1]={1,2,3,4,0,0,0,0,0,0,0}; /*Impulse Response Coefficients*/
int y[LENGHT1+LENGHT2-1];
main()
```

```
{
   int i=0,j;
   for(i=0;i<(LENGHT1+LENGHT2-1);i++)
   {
     y[i]=0;
     for(j=0;j<=i;j++)
        y[i]+=x[j]*h[i-j];
    }
 for(i=0;i<(LENGHT1+LENGHT2-1);i++)
 printf("%d\n",y[i]);
}
```

4.DFT:

```
#include<stdio.h>
#include<math.h>
int N,k,n,i;
float pi=3.1416,sumre=0, sumim=0,out_real[8]={0.0}, out_imag[8]={0.0};
int x[32];
void main(void)
{
  printf(" enter the length of the sequence\n");
  scanf("%d",&N);
  printf(" enter the sequence\n");
  for(i=0;i<N;i++)
    scanf("%d",&x[i]);
  for(k=0;k<N;k++)
  {
      sumre=0;
      sumim=0;
      for(n=0;n<N;n++)
      {
        sumre=sumre+x[n]* cos(2*pi*k*n/N);
        sumim=sumim-x[n]* sin(2*pi*k*n/N);
      }
    out_real[k]=sumre;
    out_imag[k]=sumim;
    printf("X([%d])=\t%f\t+\t%fi\n",k,out_real[k],out_imag[k]);
  }
}
```

5.INTERPOLATION & DECIMATION:

```
#include<stdio.h>
#include<math.h>
#define FREQ 500
int tb=2; //sampling rate multiplier for interpolated signal
float tx[128]; //to store sine(input) wave Signal
float ty[256]; // to store Interpolated signal
float x[256],y[256]; //to store decemated Signal
void main()
{
  int txx,ti,tc,td;
  int a,b,d;
  int i,j,z,g;
  int tj,tcc,tz;
  int ta=128;
  for(i=0;i<127;i++)
  {
      tx[i]=sin(2*3.14*FREQ*i/24000); //generating sine wave
      printf("%f\n",tx[i]);
  }
//Interpolation
  tc = tb - 1;
```

```
   txx = 0;
   for (ti=1;ti<=ta;ti++)
   {
        ty[ti+txx] = tx[ti];
        tcc = ti+txx;
        tz = ti;
        for (tj = 1 ; tj<=tc ;tj++)
        {
           ty[tcc+1] = 0; //adding zeros in between samples
           ti = ti+1;
           tcc = tcc+1;
        }
        txx = tcc-tz;
        ti = ti-tc;
    }
    td = ta*tb; //Length of interpolated signal
    for(ti=1;ti<=td;ti++)
    {
        printf("\n The Value of output ty[%d]=%f",ti,ty[ti]);
    }
//Decimation
    b=2; //Sampling rate divider for decimated signal
    j=1;
    for (g=1;g<=128;g++)
    {
       y[g] = ty[j];
       j = j+b;
       printf("%f\n",y[g]);
 }
}
```

6.FSK:

```
#include<stdio.h>
#include<math.h>
#define TIME 336 //length of FSK signal
#define PI 3.14
float fh[TIME],fl[TIME],FSK[TIME];
int input_string[8],scale_data[TIME];
void main()
{
   int i,j,k,l;
   printf("\nEnter your digital data string in the form of 1 & 0s\n");
   for(i=0;i<8;i++) //data in binary format
      scanf("%d",&input_string[i]);
      k=0;
      for(k=0;k<TIME;k++)
      {
        for(j=0;j<8;j++)
       {
         for(i=0;i<21;i++)
         {
           scale_data[k]=input_string[j]; // Scaling input data
           k++;
         }
       }
      }
     for(i=0;i<TIME;i++)
     {
       fh[i]=sin(2*PI*2000*i/10000);//high frequency
       fl[i]=sin(2*PI*1000*i/10000);//low frequency
     }
//assigning high frequency to bit 1
     k=0;
     for(l=0;k<TIME;l++)
     {
       for(i=0;i<8;i++)
       {
         if(input_string[i]==1)
         {
            for(j=0;j<21;j++)
           {
             FSK[k]=fh[j];
             k++;
           }
         }
      else
//assigning low frequency to bit 0
    {
       if(input_string[i]==0)
      {
         for(j=0;j<21;j++)
        {
          FSK[k]=fl[j];
          k++;
       }}}}}}
```

7.ENERGY OF SAMPLES:

```
#include<stdio.h>
int main()
{
   int num,i,j,x[32];
   long int sum=0;
```

```
  printf("\nEnter the number of samples: ");
  scanf("%d",&num);
  printf("\nEnter samples: ");
  for(j=0;j<num;j++)
    scanf("%d",&x[j]);
  for(i=0;i<=num;i++)
   {
    sum+=x[i]*x[i];
   }
  printf("\n the energy of above samples is\n %d",sum);
  return 0;
}
```

8.POWER OF SAMPLES:

```
#include<stdio.h>
int main(){
  int num,i,j,x[32];
  float num1;
  long int sum=0;
  printf("\nEnter the number of samples: ");
  scanf("%d",&num);
  printf("\nEnter samples: ");
  for(j=0;j<num;j++)
     scanf("%d",&x[j]);
  for(i=0;i<=num;i++)
  {
     sum+=x[i]*x[i];
  }
  num=num*2;
  num++;
  num1 = sum / (float) num;
```

```
   printf("\n the Average power of above samples is\n %.2f",num1);
   return 0;
}
```

9.CIRCULAR CONVOLUTION:

```
#include<stdio.h>

   int m,n,x[30],y[30],h[30],i,j,temp[30],k,x2[30],a[30];
   void main()
{
   printf("Enter the length of the first sequence \n");
   scanf("%d",&m);
   printf("Enter the length of second sequence \n ");
   scanf("%d",&n);
   printf("Enter the first sequence \n");
   for(i=0;i<m;i++)
       scanf("%d",&x[i]);

    printf("Enter the second sequence \n");
    for(j=0;j<n;j++)
       scanf("%d",&h[j]);

    if(m-n!=0)
    {
        if(m>n)
        {
            for(i=n;i<m;i++)
                h[i] = 0;
                n = m;
        }
            for(i=m;i<n;i++)
                x[i] = 0;
                m = n;
            }
            y[0] = 0;
            a[0] = h[0];
            for(j=1;j<n;j++)
                a[j] = h[n-j];

            for(i=0;i<n;i++)
               y[0] += x[i]*a[i];
               for(k=1;k<n;k++)
               {
                   y[k] = 0;
                   for(j=1;j<n;j++)
                      x2[j] = a[j-1];
                   x2[0] = a[n-1];
                   for(i=0;i<n;i++)
                   {
                       a[i] = x2[i];
                       y[k] += x[i]*x2[i];
                   }
               }

            printf("The circular convolution is \n");
            for(i=0;i<n;i++)
               printf("%d \t", y[i]);
        }
```

10.LOOP BACK:

```
#include "L138_LCDK_aic3106_init.h"
```

```
interrupt void interrupt4(void) // interrupt service routine
{
   uint32_t sample;
   sample = input_sample(); // read L + R samples from ADC
   output_sample(sample); // write L + R samples to DAC
 return;
}
int main(void)
{

L138_initialise_intr(FS_48000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_LINE_INPUT);
   while(1);
}
```

11. ECO-EFFECT:

```
#include "L138_LCDK_aic3106_init.h"
#define GAIN 0.6
#define BUF_SIZE 16000
  int16_t input,output,delayed;
  int16_t buffer[BUF_SIZE];
  int i = 0;
interrupt void interrupt4(void) // interrupt service routine
{
  input = input_left_sample();
  delayed = buffer[i];
  output = delayed + input;
  buffer[i] = input + delayed*GAIN;
  i = (i+1)%BUF_SIZE;
  output_left_sample(output);
  return;
}
int main(void)
{
  int i;

  for (i=0 ; i<BUF_SIZE ; i++)
 {
    buffer[i] = 0;
 }

L138_initialise_intr(FS_48000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_MIC_INPUT);
   while(1);
}
```

12. DELAY:

```
#include "L138_LCDK_aic3106_init.h"
#define BUF_SIZE 24000
uint16_t input,output,delayed;
uint16_t buffer[BUF_SIZE];
int i = 0;
interrupt void interrupt4(void) // interrupt service routine
{
  input = input_left_sample();
  delayed = buffer[i];
  output = delayed + input;
  buffer[i] = input;
  i = (i+1)%BUF_SIZE;
  output_left_sample(output);
  return;
}
int main(void)
```

```
{
  int i;
  for (i=0 ; i<BUF_SIZE ; i++)
   {
     buffer[i] = 0;
   }

L138_initialise_intr(FS_48000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_MIC_INPUT);
  while(1);
}
```

SINE WAVE:

```
#include "L138_LCDK_aic3106_init.h"
#include "math.h"
#define SAMPLING_FREQ 8000
#define PI 3.14159265358979
float frequency = 1000.0;
float amplitude = 20000.0;
float theta_increment;
float theta = 0.0;
interrupt void interrupt4(void) // interrupt service routine
{
  theta_increment = 2*PI*frequency/SAMPLING_FREQ;
  theta += theta_increment;
  if (theta > 2*PI) theta -= 2*PI;
  output_left_sample((int16_t)(amplitude*sin(theta)));
  return;
}
int main(void)
{
```

```
L138_initialise_intr(FS_8000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_LINE_INPUT);
   while(1);
}
```

SQUARE WAVE:

```
#include "L138_LCDK_aic3106_init.h"
#define LOOPLENGTH 64
int16_t square_table[LOOPLENGTH] =
{10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000,
10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000,
10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000,
10000, 10000, 10000, 10000, 10000, 10000, 10000, 10000,
-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,
-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,
-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,
-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000};
int16_t loopindex = 0;
interrupt void interrupt4(void) // interrupt service routine
{
  output_left_sample(square_table[loopindex++]);
  if (loopindex >= LOOPLENGTH)
  loopindex = 0;
  return;
}
int main(void)
{

L138_initialise_intr(FS_8000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_LINE_INPUT);
  while(1);
}
```

Ramp Wave Generation:

```
#include "L138_LCDK_aic3106_init.h"
#define LOOPLENGTH 64
int16_t output = 0;
interrupt void interrupt4(void) // interrupt service routine
{
   output_left_sample(output); // output to L DAC
   output += 2000; // increment output value
   if (output >= 30000) // if peak is reached
       output = -30000; // reinitialize
   return;
}
int main(void)
{


L138_initialise_intr(FS_8000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_LINE_INPUT);
  while(1);
}
```