1.Random Wave:

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#define PI 3.14
#define PTS 128

float X[PTS];
float Y[PTS];
float Z[PTS];
float N[PTS];

int main() {
	int i;

	for (i = 0; i < PTS; i++) {
	X[i] = sin(2 * PI * i * 20 / 128.0);
	Y[i] = 0.0;
	N[i] = X[i] + rand() * 10;
	}

	// Rest of your code...

	//return 0;
}
```

2. Differential equation

```
#include <stdio.h>
#include <math.h>
#define FREQ 400
float y[3] = {0, 0, 0};
float x[3] = {0, 0, 0};
float z[128], m[128], n[128], p[128];

int main() {
	int i, j;
	float a[3] = {0.072231, 0.144462, 0.072231};
	float b[3] = {1.000000, -1.109229, 0.398152};

	for (i = 0; i < 128; i++) {
	m[i] = sin(2 * 3.14 * FREQ * i / 24000);
	}

	for (j = 0; j < 128; j++) {
	x[0] = m[j];
	y[0] = (a[0] * x[0]) + (a[1] * x[1]) + (x[2] * a[2]) - (y[1] * b[1]) - (y[2] * b[2]);
	z[j] = y[0];
	y[2] = y[1];
	y[1] = y[0];
	x[2] = x[1];
	x[1] = x[0];
	}

	// Rest of your code...
	//return 0;
}
```

3.Power

```
#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("Enter samples:\n");
        for (j = 0; j < num; j++) {
        scanf("%d", &x[j]);
        }

        for (i = 0; i < num; i++) {
        sum += x[i] * x[i];
        }

        num *= 2;
         num++;
        num1 = sum / (float)num;

        printf("\nThe Average power of above samples is %.2f\n", num1);

        return 0;
}
```

4. Energy

```
#include<stdio.h>

int main() {
        int num, i, j, x[32];
        long int sum = 0;

        printf("Enter the number of samples: ");
        scanf("%d", &num);

        printf("Enter samples:\n");
        for (j = 0; j < num; j++) {
        scanf("%d", &x[j]);
        }

        for (i = 0; i < num; i++) {
        sum += x[i] * x[i];
        }

        printf("The energy of above samples is %ld\n", sum);

      return 0;
}
```

7. Circular Convolution. X1(n) = {1,1,2,1} and X2(n)= {1,2,3,4}

```
#include<stdio.h>
int m, n, x[30], h[30], y[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 the 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) {
        // Zero-padding to make both sequences of equal length
        int max_len = (m > n) ? m : n;
        for (i = m; i < max_len; i++)
        x[i] = 0;
        for (j = n; j < max_len; j++)
        h[j] = 0;
        m = n = max_len;
        }

        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];
        }
        }

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

        printf("\n");
}
```

5. DFT X(n)= {1,1,1,1,1,1,0,0}

```
#include<stdio.h>
#include<math.h>
int N, k, n, i;
float pi = 3.1416, sumre = 0, sumim = 0;
float out_real[32] = {0.0}, out_imag[32] = {0.0};
int x[32];

int 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 += x[n] * cos(2 * pi * k * n / N);
        sumim -= x[n] * sin(2 * pi * k * n / N); // Fixed a typo here (Sin to sin)
        }

        out_real[k] = sumre;
        out_imag[k] = sumim;
        printf("X([%d]) = %.4f + %.4fi\n", k, out_real[k], out_imag[k]);
        }

        return 0;
}
```

6.Autocorrelation

```
#include<stdio.h>
#define PTS 4

int x[PTS], y[PTS], atoc[PTS], n, k;

int main() {
        int i, sum;

        printf("Enter the first sequence\n");
        for (i = 0; i < PTS; i++)
        scanf("%d", &x[i]);

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

        for (n = 0; n < PTS; n++) {
        sum = 0;
        for (k = 0; k < PTS - n; k++) {
        sum += (x[k] * y[k + n]);  // Changed to cross-correlation
        }
        atoc[n] = sum;
        }

        printf("Autocorrelation result:\n");
        for (i = 0; i < PTS; i++)
        printf("%d\n", atoc[i]);

        return 0;
}
```

7, Linear Convolution $X_1(n) = \{2, 3, 4\}$ & $X_2(n) = \{1, 2, 1\}$

o/p-y[n]={2,7,12,11,4}

```
#include <stdio.h>
#define N1 3
#define N2 3
#define MAX_RESULT_LENGTH (N1 + N2 - 1)

int x1[N1] = {2, 3, 4};
int x2[N2] = {1, 2, 1};
int result[MAX_RESULT_LENGTH];

void convolution() {
        for (int i = 0; i < MAX_RESULT_LENGTH; i++) {
        result[i] = 0;
        for (int j = 0; j <= i; j++) {
        if (j < N1 && (i - j) < N2) {
        result[i] += x1[j] * x2[i - j];
        }
        }
        }
}

int main() {
        // Perform convolution
        convolution();

        // Display the result
        printf("Result of convolution:\n");
        for (int i = 0; i < MAX_RESULT_LENGTH; i++) {
        printf("%d ", result[i]);
        }
        printf("\n");

        return 0;
}
```

Ramp:-

```
#include "L138_LCDK_aic3106_init.h"

#define LOOPLENGTH 64
int16_t output = 0;

interrupt void interrupt4(void)
{
	output_left_sample(output);
	output += 2000;

	if (output >= 30000)
	output = -30000;

	return;
}

int main(void)
{
	L138_initialise_intr(FS_8000_HZ, ADC_GAIN_0DB, DAC_ATTEN_0DB, LCDK_LINE_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)
{
	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)
	{
	// Your main loop code (if needed)
	}
return 0;
}
```

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,-10
000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-10000,-100
00}}; // Square wave table
int16_t loopindex = 0;
interrupt void interrupt4(void)
{
        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);
}
```

Flanging effect:-

```
#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)
{
        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;

        // Initialize buffer to zero
        for (i = 0; i < BUF_SIZE; i++)
        {
        buffer[i] = 0;
        }

        // Initialize and configure the LCDK for audio input
        L138_initialise_intr(FS_48000_HZ, ADC_GAIN_0DB, DAC_ATTEN_0DB, LCDK_MIC_INPUT);
while (1);

}
```

10. Headphone

```
#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)
{
	input = input_left_sample();
	delayed = buffer[i];
	output = delayed + input;
	buffer[i] = input + delayed * GAIN;
	i = (i + 1) % BUF_SIZE;

	output_left_sample(output);
	output_right_sample(output);  // Send the processed signal to the right channel

	return;
}

int main(void)
{
	int i;

	// Initialize buffer to zero
	for (i = 0; i < BUF_SIZE; i++)
	{
	buffer[i] = 0;
	}

	// Initialize and configure the LCDK for audio input
	L138_initialise_intr(FS_48000_HZ, ADC_GAIN_0DB, DAC_ATTEN_0DB, LCDK_MIC_INPUT);

	while (1)
	{
	// Infinite loop to keep the program running
	}
}
```