Using your Arduino Uno and the arduinoFFT.h library, write a short program that outputs the dominant frequency between 0 and 5 Hz, 5 and 12 Hz, 12 and 24Hz, and between 24 and 36Hz, and outputs their corresponding magnitudes given the discrete time data provided in the following array. Assume this data was collected at a sample rate of 128hz for 1 second from analog pin 0.

```
int dataArray[128] = {10, 25, 10, 50, 10, 5, 10, 50, 10, 25, 10, 50, 10, 5, 10, 50, 75, 25, 10, 50, 10, 5, 10, 50,
              10, 25, 10, 50, 10, 5, 10, 110, 75, 25, 10, 50, 10, 5, 10, 50, 10, 25, 10, 50, 10, 5, 10, 50, 75, 25,
              10, 50, 10, 5, 10, 50, 10, 25, 10, 50, 10, 5, 10, 110, 75, 25, 10, 50, 10, 5, 10, 50, 10, 25, 10, 50,
              10, 5, 10, 50, 75, 25, 10, 50, 10, 25, 10, 50, 10, 5, 10, 50, 10, 25, 10, 110, 75, 5, 10, 50, 10, 25,
              10, 50, 10, 5, 10, 50, 10, 25, 10, 50, 75, 5, 10, 50, 10, 25, 10, 50, 10, 5, 10, 50, 10, 25, 10, 128
             };
```

You are free to use a different library or even develop all your own code if you wish.

```
int dataArray[128] = {10, 25, 10, 50, 10, 5, 10, 50, 10, 25, 10, 50, 10, 5, 10, 50, 75, 25, 10,
50, 10, 5, 10, 50,
                      10, 25, 10, 50, 10, 5, 10, 110, 75, 25, 10, 50, 10, 5, 10, 50, 10, 25,
10, 50, 10, 5, 10, 50, 75, 25,
                      10, 50, 10, 5, 10, 50, 10, 25, 10, 50, 10, 5, 10, 110, 75, 25, 10, 50,
10, 5, 10, 50, 10, 25, 10, 50,
                      10, 5, 10, 50, 75, 25, 10, 50, 10, 25, 10, 50, 10, 5, 10, 50, 10, 25, 10,
110, 75, 5, 10, 50, 10, 25,
                      10, 50, 10, 5, 10, 50, 10, 25, 10, 50, 75, 5, 10, 50, 10, 25, 10, 50, 10,
5, 10, 50, 10, 25, 10, 128
                     };
float f_peaks[5]; // top 5 frequencies peaks in descending order
//----------------------------------------------------------------------------//


void setup()
        {
        Serial.begin(250000);
        }


void loop() {

//-----------------------------FFT Function----------------------------------------------//

float FFT(int in[],int N,float Frequency)
{
unsigned int data[13]={1,2,4,8,16,32,64,128,256,512,1024,2048};
int a,c1,f,o,x;
a=N;

      for(int i=0;i<12;i++)                 //calculating the levels
         { if(data[i]<=a){o=i;} }

int in_ps[data[o]]={};     //input for sequencing
float out_r[data[o]]={};   //real part of transform
float out_im[data[o]]={};  //imaginory part of transform

x=0;
      for(int b=0;b<o;b++)                     // bit reversal
         {
          c1=data[b];
          f=data[o]/(c1+c1);
                for(int j=0;j<c1;j++)
                    {
                     x=x+1;
                     in_ps[x]=in_ps[j]+f;
                    }
         }


      for(int i=0;i<data[o];i++)            // update input array as per bit reverse order
         {
          if(in_ps[i]<a)
          {out_r[i]=in[in_ps[i]];}
          if(in_ps[i]>a)
          {out_r[i]=in[in_ps[i]-a];}
         }


int i10,i11,n1;
float e,c,s,tr,ti;

    for(int i=0;i<o;i++)                                    //fft
    {
     i10=data[i];              // overall values of sine/cosine  :
     i11=data[o]/data[i+1];    // loop with similar sine cosine:
     e=360/data[i+1];
     e=0-e;
     n1=0;

          for(int j=0;j<i10;j++)
          {
          c=cosine(e*j);
          s=sine(e*j);
          n1=j;

                for(int k=0;k<i11;k++)
                 {
                 tr=c*out_r[i10+n1]-s*out_im[i10+n1];
                 ti=s*out_r[i10+n1]+c*out_im[i10+n1];

                 out_r[n1+i10]=out_r[n1]-tr;
                 out_r[n1]=out_r[n1]+tr;

                 out_im[n1+i10]=out_im[n1]-ti;
                 out_im[n1]=out_im[n1]+ti;

                 n1=n1+i10+i10;
                  }
             }
     }

/*
for(int i=0;i<data[o];i++)
{
Serial.print(out_r[i]);
Serial.print("\t");                                      // un comment to print RAW o/p
Serial.print(out_im[i]); Serial.println("i");
}
*/


//---> here onward out_r contains amplitude and our_in conntains frequency (Hz)
    for(int i=0;i<data[o-1];i++)               // getting amplitude from compex number
        {
         out_r[i]=sqrt(out_r[i]*out_r[i]+out_im[i]*out_im[i]); // to  increase the speed delete
sqrt
         out_im[i]=i*Frequency/N;
         /*
         Serial.print(out_im[i]); Serial.print("Hz");
         Serial.print("\t");                            // un comment to print freuency bin
         Serial.println(out_r[i]);
         */
        }




x=0;       // peak detection
   for(int i=1;i<data[o-1]-1;i++)
      {
      if(out_r[i]>out_r[i-1] && out_r[i]>out_r[i+1])
      {in_ps[x]=i;    //in_ps array used for storage of peak number
      x=x+1;}
      }


s=0;
c=0;
    for(int i=0;i<x;i++)             // re arraange as per magnitude
    {
        for(int j=c;j<x;j++)
        {
            if(out_r[in_ps[i]]<out_r[in_ps[j]])
                {s=in_ps[i];
                in_ps[i]=in_ps[j];
                in_ps[j]=s;}
        }
    c=c+1;
    }



    for(int i=0;i<5;i++)     // updating f_peak array (global variable)with descending order
    {
    f_peaks[i]=out_im[in_ps[i]];
    }



}


float sine(int i)
```

```
{
  int j=i;
  float out;
  while(j<0){j=j+360;}
  while(j>360){j=j-360;}
  if(j>-1   && j<91){out= sine_data[j];}
  else if(j>90  && j<181){out= sine_data[180-j];}
  else if(j>180 && j<271){out= -sine_data[j-180];}
  else if(j>270 && j<361){out= -sine_data[360-j];}
  return (out/255);
}

float cosine(int i)
{
  int j=i;
  float out;
  while(j<0){j=j+360;}
  while(j>360){j=j-360;}
  if(j>-1   && j<91){out= sine_data[90-j];}
  else if(j>90  && j<181){out= -sine_data[j-90];}
  else if(j>180 && j<271){out= -sine_data[270-j];}
  else if(j>270 && j<361){out= sine_data[j-270];}
  return (out/255);
}
```