```
% Author: Gustav Kjellberg
```

Brief summary of this function.

This function takes the input from GetMusicFeatures and post processes the data so that it may better work with our HMM.

Detailed explanation of this function.

Input:

Array-like (3xT) shape, i.e the data yielded from GetMusicFeatures.

Returns:

Array-like (1xT) shape, i.e the features.

```
function features = Postprocess(musicFeatures)
```

Let first assume that the data we obtain from `GetMusicFeatures` is normally distributed. This is a simple assumption and the hueristic might not be valid but it will hopefully enable us to easier remove nosie and identify silent sections. This will be done by verifying if a datapoint is an outlier or not.

1.Divide and parse the data

The relation between sub-notes does obviously not need to be taken as a log while the both frequency and intensity should be, as stated in the assignment.

```
frequency = log(musicFeatures(1,:));
relation = musicFeatures(2,:);
intensity = log(musicFeatures(3,:));
```

2.Let's now calculate mean and variance in order to utlize that to find outlier that thus oculd be silent or noise sections.

```
frequencyMean = mean(frequency);
frequencyStd = std(frequency);

frequencyUpperBound = frequencyMean+frequencyStd;
frequencyLowerBound = frequencyMean-frequencyStd;
frequencyNormalDist = makedist('Normal', "mu",frequencyMean, "sigma",frequencyStd);
frequencyTargetDist = fitdist(transpose(frequency), "Normal");

relationMean = mean(relation);
relationStd = std(relation);

relationUpperBound = relationMean+relationStd;
relationLowerBound = relationMean-relationStd;
relationNormalDist = makedist('Normal', "mu",relationMean, "sigma",relationStd);
relationTargetDist = fitdist(transpose(relation), 'Normal');
```

```
intensityMean = mean(intensity);
intensityStd = std(intensity);

intensityUpperBound = intensityMean+intensityStd;
intensityLowerBound = intensityMean-intensityStd;
intensityNormalDist = makedist('Normal', "mu",intensityMean, "sigma",intensityStd);
intensityTargetDist = fitdist(transpose(intensity), 'Normal');
```

Starting off by testing ony looking at outliers that are outside of the lower bounds. We never want to look at high relation as noise, since that is good, intensity is unknown to me as of now and the high frequencies might be pitches when singing... Testing to set diffrent bounds in order to see if we get a big difference in results. Changed to that the relation can't be weaker than the mean, which was good and quite reasonalbe that we are hard on that specific feature. I'm not capping intensity as that can be an increase in volume.

```
noiseOrQuite = (frequency<frequencyLowerBound)|(frequency>frequencyUpperBound)|(relatio

nonNoiseIndecies = find(frequency == 0);
```

Not to find the lower octave, let's try to find the smallest frequency value that isn't calssified as noise:

```
f_0 = min(frequency(find(noiseOrQuite == 0)));
```

Calculate features, this builds on Twelfth root of two

```
features = 12*log2(frequency/f_0);
```

Set the noisy regions to zero

```
features(find(noiseOrQuite==1)) = 0;
```

Transform the contignoues values into scalars:

```
features = round(features);



end
```