

众所周知的蜂鸣器

时间限制 : 10 Sec

空间限制 : 512 Mb

众所周知，Boshi是个喜欢弹琴的妹子。

TA经常在世界各地弹琴，比如APIO/CTSC赛场。

众所周知，Menhera是个喜欢唱歌的妹子。

TA经常在世界各地唱歌，比如各种模拟考现场。

众所周知，Rayment是个喜欢电音的妹子。

TA经常在世界各地听电音，比如Logic Gatekeeper。

众所周知，XZY是个喜欢玩蜂鸣器的弱智。

TA经常在世界各地玩蜂鸣器，然后被劝退。

最近XZY又在玩蜂鸣器了，TA都玩了三年蜂鸣器了，从初中就开始玩了，玩到现在才玩出一个不着调的《极乐净土》，这让TA很难过。

TA曾经想，如果能直接把下载下来的音乐直接转化成控制蜂鸣器的代码就好了。

控制蜂鸣器的代码长这样：

```
1 Beep(a, b);
```

表示蜂鸣器以**a**的频率振动（单位为赫兹），连续振动**b**毫秒。

TA去世界各地查有没有这样的软件，但是众所周知，没有人会像TA这么无聊，所以没有。

TA也很想自己写一个，但是因为自己太弱了所以写不出，TA甚至都不知道音频文件的储存格式。

于是TA找啊找，找到了百度，找到了阿里巴巴，找到了四十大盗，找到了腾讯，找到了鲁迅，找到了枣树，找到了新中国。。。

最终TA在Google上找到了这样一段c++代码可以读入.wav格式的音乐（你可以不用读懂它在干什么的，等下XZY会向你说明的）：

```
1 #include <unistd.h>
2 #include <stdio.h>
3 #include <string.h>
4 #include <stdlib.h>
5 #include <limits.h>
6
7 namespace WFR // Wave File Reader
8 {
```

```

9      double Length;
10     unsigned int num_samples;
11     int low_limit  = 01;
12     int high_limit = 01;
13     int BUF[2][2000005];
14     int channels;
15     struct HEADER
16     {
17         unsigned char riff[4];           /* RIFF string */
18         unsigned int  overall_size;      /* overall size of file in bytes */
19         unsigned char wave[4];          /* WAVE string */
20         unsigned char fmt_chunk_marker[4]; /* fmt string with trailing null
char */
21         unsigned int  length_of_fmt;     /* length of the format data */
22         unsigned int  format_type;       /* format type. 1-PCM, 3- IEEE
float, 6 - 8bit A law, 7 - 8bit mu law */
23         unsigned int  channels;          /* no.of channels */
24         unsigned int  sample_rate;       /* sampling rate (blocks per second)
*/
25         unsigned int  byterate;          /* SampleRate * NumChannels *
BitsPerSample/8 */
26         unsigned int  block_align;       /* NumChannels * BitsPerSample/8 */
27         unsigned int  bits_per_sample;   /* bits per sample, 8- 8bits, 16- 16
bits etc */
28         unsigned char data_chunk_header [4]; /* DATA string or FLLR string */
29         unsigned int  data_size;         /* NumSamples * NumChannels *
BitsPerSample/8 - size of the next chunk that will be read */
30     };
31     unsigned char  buffer4[4];
32     unsigned char  buffer2[2];
33     struct HEADER header;
34
35     void Read()
36     {
37         int read = 0;
38         /* read header parts */
39         read = fread( header.riff, sizeof(header.riff), 1, stdin );
40         read = fread( buffer4, sizeof(buffer4), 1, stdin );
41         /* convert little endian to big endian 4 byte int */
42         header.overall_size = buffer4[0] | (buffer4[1] << 8) | (buffer4[2] << 16) |
(buffer4[3] << 24);
43         read = fread( header.wave, sizeof(header.wave), 1, stdin );
44         read = fread( header.fmt_chunk_marker, sizeof(header.fmt_chunk_marker), 1,
stdin );
45         read = fread( buffer4, sizeof(buffer4), 1, stdin );
46         /* convert little endian to big endian 4 byte integer */

```



```

85         low_limit    = SHRT_MIN;
86         high_limit   = SHRT_MAX;
87         break;
88     case 32:
89         low_limit    = INT_MIN;
90         high_limit   = INT_MAX;
91         break;
92     }
93     for ( i = 1; i <= num_samples; i++ )
94     {
95         read = fread( data_buffer, sizeof(data_buffer), 1, stdin );
96         if ( read == 1 )
97         {
98             /* dump the data read */
99             unsigned int    xchannels    = 0;
100             int             data_in_channel = 0;
101             for ( xchannels = 0; xchannels < header.channels;
xchannels++ )
102             {
103                 /* convert data from little endian to big endian based
on bytes in each channel sample */
104                 if ( bytes_in_each_channel == 4 )
105                 {
106                     data_in_channel = data_buffer[0] |
107                                     (data_buffer[1] << 8) |
108                                     (data_buffer[2] << 16) |
109                                     (data_buffer[3] << 24);
110                 }
111                 else if ( bytes_in_each_channel == 2 )
112                 {
113                     data_in_channel = data_buffer[0] |
114                                     (data_buffer[1] << 8);
115                 }
116                 else if ( bytes_in_each_channel == 1 )
117                 {
118                     data_in_channel = data_buffer[0];
119                 }
120                 BUF[xchannels][i] = data_in_channel;
121                 /* check if value was in range */
122                 if ( data_in_channel < low_limit || data_in_channel >
high_limit )
123                     printf( "***value out of range\n" );
124             }
125         }
126     else
127     {

```

```

128         printf( "Error reading file. %d bytes\n", read );
129         break;
130     }
131 } /*      for (i =1; i <= num_samples; i++) { */
132     } /*      if (size_is_correct) { */
133 } /*      if (header.format_type == 1) { */
134     else puts( "Your Wave File isn't in PCM Format!" );
135 }
136 }

```

于是TA就能读入枣树，读入音乐了，TA找到了人生的曙光，人民的希望，宇宙的奥秘，哲学的真理。

可是TA还是不会把音乐转化成Beep，于是向你请教，并且告诉了你：

上面那段代码能这样调用：

- WFR::Read(), 无返回值，表示把.wav文件读入到WFR::BUF数组里
- WFR::Length, 双精度小数型，表示读入的歌曲时长，单位为秒
- WFR::num_samples, 无符号整型，表示整首歌的采样点个数
- WFR::low_limit, 整形，表示振幅的下界（一般你是用不到的，这个跟采样位宽有关）
- WFR::high_limit, 同上，表示振幅的上界（这里注意一下，振幅可以为负数，表示当前相位在初相位下方）
- WFR::BUF[i][j], 整形，表示声道*i*的第*j*个采样点的振幅，无单位，*i*从0开始标号，*j*从1开始标号
- WFR::channels, 整形，表示声道个数

上面的那段代码你可以自己随意修改。

如果你觉得上面那个namespace太垃圾了自己写一个也是完全没有问题的。

当XZY讲完这些以后，爱弹琴的Boshi终于忍不住了，告诉你：

实际上XZY的歌曲都很垃圾，最多只有两个声道，有的只有一个声道。

所以如果有两个声道的话你可以直接取`BUF[0][j]`和`BUF[1][j]`的算术平均值作为第 j 个采样点的振幅，这样就能把两个声道合成为一个声道了。

而且如果你把一个声道的振幅数组画成柱状图，你会发现它其实就是声音信号的“时域图”

对于单个音符，它的频率是单一的，固定的。

但是音乐里一般都有干扰，比如杂音。

XZY应该只是想要你把杂音去掉，再把最明显的频率给Beep出来而已。

取出最明显的频率这种操作，估计要在“频域图”上动手脚了！

Boshi讲完了这些，你恍然大悟，连连点头，欢天喜地，载兴载舞，僮仆欢迎，稚子候门，动手动脚，牛逼哄哄。

这时候，XZY又说：

其实我的音乐都很简单的，杂音都很少，在底下的提示里我会写好每个歌曲的特点的！

于是Boshi把XZY的歌都听了一遍，发现不是很牛逼，于是现场演奏了一首《两只老虎》，XZY赶紧收藏了。

这时候一直躲在旁边的Menhera坐不住了：Menhera天下第一！于是做现场的表演，演唱了一首《?♂郝&@瀚3#鸽》，在座的小朋友们都被TA精湛的演唱功底惊呆了，纷纷高兴地戳破了耳膜，XZY也欢欣鼓舞一路火花加闪电地把美妙的歌声录了下来，赶紧收藏了。

屏幕前聪明的你，能否帮助聋了的XZY和大家完成这个任务呢？

输入格式

一个 **.wav** 格式的文件，保证是 **pcm** 格式的。

另外：

此题需要开文件，从 **wave.in** 读入，输出到 **wave.out**。

请使用二进制方式打开文件，例如在 `int main()` 的开始部分添加：

```
freopen("wave.in", "rb", stdin), freopen("wave.out", "w", stdout);
```

如果不用 `"rb"` 方式打开文件可能会导致 **Error reading file**（本机也是！）！ 请注意！

输出格式

若干行，每行两个非负整数 a 和 b ，表示 **Beep(a , b)**；

样例

请直接下载数据查看。

数据范围与提示

首先

输入文件是可以直接听的，下载下来以后把后缀名改为 **.wav**，用音乐播放器打开就能听了。

~~你乐意用这个方法打表我很资磁~~

特殊问题

1. 如果某一段时间原歌曲的声音很小，不足以判断出最明显的频率，请以前一段时间频率作为这一段时间的频率输出（若在歌曲开头处则频率直接设为0）。
2. 由于蜂鸣器都很菜，所以所有连续时长小于**50**毫秒的Beep命令都无法发声（实际上如果运行的话会很奇怪，有的能发声有的不能），对于这种Beep命令XZY将视其Beep频率为0（实际上就变成了Sleep了）

判断答案正误

XZY写了个SPJ来判断你的输出是否正确

SPJ会依次模拟执行你的**Beep**命令，如果出现以下情况则直接判0分：

- 命令个数超过 10^5
- 命令的总共执行时间（即播放时间）超过了 6×10^5 毫秒

如果上面两条都没有发生，则SPJ会用该测试点的答案文件进行判分：

- 对于某一毫秒，如果两者的输出频率之差的绝对值小于25（单位赫兹），或者标准答案这一毫秒是没有声音的（频率为0），则判这一毫秒是正确的
- 对于整首歌曲，如果有40%及以上的时间是正确的（正确率高于40%），则给满分
- 否则给出“满分×正确率÷40%”，并且分数向下取整

数据范围

歌曲时长最大为1分22秒

采样点数不超过 2×10^6 个

其他基本不用操心

更详细的见下方“歌曲特性”

歌曲特性

数据编号	歌曲名称	特性
1	两只老虎	由蜂鸣器演奏，XZY用老人机录音
2	小酒窝	由蜂鸣器演奏，XZY用老人机录音
3	极乐净土	由蜂鸣器演奏，XZY用老人机录音
4	两只老虎	Boshi演奏，XZY用老人机录音
5	fc坦克大战BGM	红白机风格，网易云下载，时长4秒
6	马里奥地上关BGM	红白机风格，XZY用老人机录音
7	极乐净土	红白机风格，网易云下载
8	斗地主	红白机风格，网易云下载
9	Only My Railgun	红白机风格，网易云下载
10	? ♂ 郝&@瀚3#鸽	Warning, nuclear missile launched!

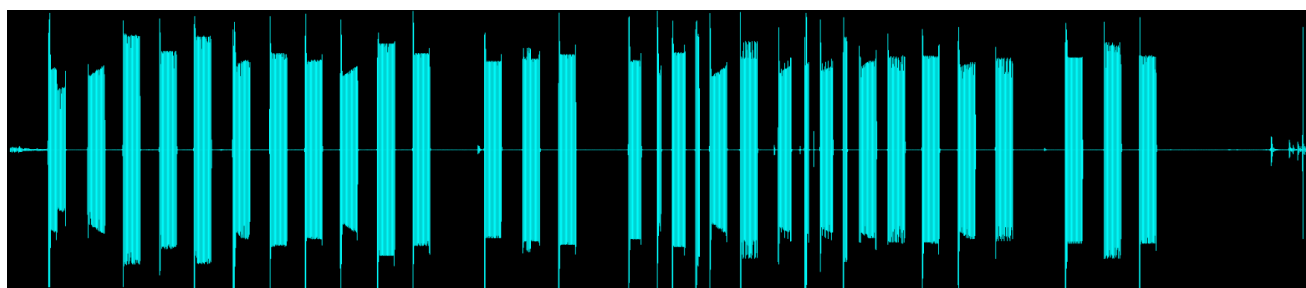
XZY的老人机型号是Nokia 216，单声道，采样率8000（采样率就是每秒的采样点数目）

网易云音乐下载的都是双声道，采样率不会高于11025

相关知识

[傅里叶分析之掐死教程](#)

第一组数据（两只老虎蜂鸣器版）的时域谱（横坐标对应时间，纵坐标对应振幅）：



第一组数据的频域谱（横坐标对应频率，纵坐标对应振幅，高峰表示突出的频率）：

