Le principe de ce projet, est d'écrire un programme qui va venir jouer une pièce musicale. Cette pièce sera écrite dans un fichier texte, et sera écrit de cette manière :

C 400

C 400

G 400

G 400

A 400

A 400

G 400

W 500

F 400

F 400

E 400

E 400

D 400

D 400

C 400

La lettre correspond à une note (en notation américaine) et le nombre au temps joué avec cette note (en milliseconde). La lettre W correspond à un temps d'attente.

Pour pouvoir produire le son, on utilisera une fonction système :

```
BOOL Beep(DWORD dwFreq, DWORD dwDuration);
```

Cette fonction produit un son de fréquence dwFreq (en Hz) pour une durée dwDuration (en milliseconde). Elle renvoie true si tout s'est bien passé, false sinon.

Pour le temps d'arrêt, on utilisera la fonction système :

```
void Sleep(DWORD dwMilliseconds);
```

Cette fonction met en pause le programme pendant dwMilliseconds.

Les fonctions systèmes sont une collection de librairie proposés par le système d'exploitation (ici windows) pour interagir avec le système. C'est-à-dire, le gestionnaire de fichier, la gestion de l'audio et de la vidéo, les fonctions réseaux, etc… Cette collection s'appelle WIN32 API. Tu pourras retrouver toutes les fonctionnalités proposées dans sa documentation ici.
Bien entendu, un programme qui utilise WIN32 ne pourras pas être utilisé sur mac ou linux (pas dérangeant dans notre cas), si l'on veut que le programme soit portable et utilisable sur tous les systèmes d'exploitation, on se tournera vers des librairie libre, type openGL, openAL, Boost, QT, etc.

1.

Commençons par écrire la fonction bool playnote (char note, int time). Elle seras composé d'un switch pour savoir qu'elle note est passé en paramètre (A , B, C, D, E, F, G ou W), puis de faire le bip correspondant, ou de mettre en attente le programme pour la durée passé en paramètre. Si une autre lettre est passé en paramètre, la fonction renvoi false, true sinon. Voici la fréquence correspondante à chaque note :

A	2750
B	3087
C	1637
D	1835
E	2060
F	2183
G	2450

2.

Nous allons maintenant écrire un parser. Un parser est un programme qui va venir analyser un texte, et appliquer le traitement. Le but ici sera de lire le fichier, ligne par ligne, et de jouer la note correspondante pendant la durée correspondante grâce a la fonction playnote. A chaque appel de playnote on vérifie bien que le retour est true. Voici un exemple de lecture de fichier :

```
#include <iostream>
#include <fstream>
using namespace std;

int main () {
   ifstream file;
   file.open ("program.txt");
    if (!file.is_open()){
        return 0;
    }

    string word;
    while (file >> word)
    {
        Cout << word << endl;
    }
}
```

Les choses à voir ici sont d'abord, les include. Il faut y mettre la librairie fstream, qui gère la lecture et l'écriture de fichier. Le fichier est un type à part entière, ifstream pour un fichier d'input (en lecture), ou ofstream pour un fichier d'output (en écriture). Lorsque l'on ouvre un fichier, on donne à la fonction open un chemin relatif. La racine du chemin est l'emplacement de ton fichier .exe .
L'opérateur << et >> servent en réalité à écrire ou extraire d'un flux. Cin et cout sont donc deux flux, qui correspondent aux flux du terminale. Pour lire le fichier on fera , file >> word1 >> word2 ; et pour écrire dans un fichier on fera file << word1 << word2 ;
Ecris la fonction void parse(string fileName) qui va venir lire le fichier passé en paramètre et jouer la partition.

3.

Il nous reste plus qu'à écrire le main. Pour ce projet, on utilisera les arguments de main. Il existe deux signature possible pour le main. Int main() et int main(int argc, char *argv[]). La seconde signature permet de faire passé des paramètres a notre programme. C'est particulièrement utilisé dans les programmes qui se manipule avec la console, comme git par exemple. Lorsque tu fait git commit -

m «coucou» , tu lance le programme git avec 3 arguments, commit, -m, et «coucou». Dans la signature, argc correspond au nombre de paramètre, argv est un tableau qui va contenir les valeurs des paramètres, le nom du programme est aussi considéré comme un paramètre. Dans notre exemple avec git, argc serait égale à 4, argv[0] à git, argv[1] à commit, argv[2] à -m et argv[3] à «coucou».
Pour notre projet, on passera le chemin de la partition dans les paramètres du main. Pour lancer notre programme, on le fera depuis un terminal, et de cette manière :

```
Beepsong partition.txt
```