```
// 演習課題5: 圧電スピーカー使用，適当な音階メロディの作成

#include <math.h>  // 算術用ライブラリ

// defineはコンパイル時にマクロ変換される
#define BZ 9       // 圧電スピーカー接続ピン

/**
 * 関数名: setup
 * 引数: なし
 * 処理: 各初期設定のため初回のみ実行
 * 返り値: なし
 */
void setup () {
    // put your setup code here, to run once:

    pinMode ( BZ, OUTPUT );  // 圧電スピーカー接続ピン
    Serial.begin ( 9600 );   // シリアル通信の初期化
}

/**
 * 関数名: loop
 * 引数: なし
 * 処理: 無限ループ．適当な音階メロディを再生する
 * 返り値: なし
 */
void loop () {
    // put your main code here, to run repeatedly:
    // tone関数は3, 11番ピンの出力を妨げる

    tyarumera ();    // チャルメラ再生関数
    delay ( 3000 );  // 次の歌の間
    doremiSong ();   // ドレミの歌再生関数
    delay ( 3000 );  // 次の歌の間
}

/**
 * 関数名: tyarumera(自作関数)
 * 引数: なし
 * 処理: チャルメラの音楽を流す
 * 返り血: なし
 */
void tyarumera () {
    // tone関数は3, 11番ピンの出力を妨げる

    // 音の長さを格納する変数
    int shortTone = 300;      // 短い音[ms]
```

```
    int middleTone = 500;     // 中間長の音[ms]
    int longTone = 800;       // 長い音[ms]
    int veryLongTone = 1000;  // とても長い音[ms]

    // 同メロディ2回
    for ( int i = 0; i < 2; i++ ) {
        tone ( BZ, scale2Hz ( "do" ), middleTone );
        delay ( middleTone );
        tone ( BZ, scale2Hz ( "re" ), shortTone );
        delay ( shortTone );
        tone ( BZ, scale2Hz ( "mi" ), veryLongTone );
        delay ( veryLongTone );
        tone (BZ, scale2Hz ( "re" ), shortTone );
        delay ( shortTone );
        tone ( BZ, scale2Hz ( "do" ), middleTone );
        delay ( veryLongTone );
    }
    // 最後
    tone ( BZ, scale2Hz ( "do" ), middleTone );
    delay ( middleTone );
    tone ( BZ, scale2Hz ( "re" ), shortTone );
    delay ( shortTone );
    tone ( BZ, scale2Hz ( "mi" ), middleTone );
    delay ( middleTone );
    tone ( BZ, scale2Hz ( "re" ), 1500 );
}

/**
 * 関数名: doremiSong(自作関数)
 * 引数: なし
 * 処理: ドレミの歌を流す
 * 返り血: なし
 */
void doremiSong () {
    int veryShortTone = 150;  // とても短い音[ms]
    int shortTone = 300;      // 短い音[ms]
    int longTone = 800;       // 長い音[ms]
    int veryLongTone = 1000;  // とても長い音[ms]

    tone ( BZ, scale2Hz ( "do" ), longTone );
    delay ( longTone );
    tone ( BZ, scale2Hz ( "re" ), shortTone );
    delay ( shortTone );
    tone ( BZ, scale2Hz ( "mi" ), longTone );
    delay ( longTone );
    tone ( BZ, scale2Hz ( "do" ), shortTone );
    delay ( shortTone );
    tone ( BZ, scale2Hz ( "mi" ), longTone );
    delay ( longTone );
    tone ( BZ, scale2Hz ( "do" ), shortTone );
    delay ( shortTone );
```

```
tone ( BZ, scale2Hz ( "mi" ), longTone );
delay ( longTone + 150 );

tone ( BZ, scale2Hz ( "re" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "mi" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "fa" ), shortTone );
delay ( shortTone + 30);
tone ( BZ, scale2Hz ( "fa" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "mi" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "re" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "fa" ), longTone );
delay ( longTone + 150 );

tone ( BZ, scale2Hz ( "mi" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "fa" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "so" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "mi" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "so" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "mi" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "so" ), longTone );
delay ( longTone + 150 );

tone ( BZ, scale2Hz ( "fa" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "so" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "ra" ), shortTone );
delay ( shortTone + 30);
tone ( BZ, scale2Hz ( "ra" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "so" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "fa" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "ra" ), longTone );
delay ( longTone + 150 );

tone ( BZ, scale2Hz ( "so" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "do" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "re" ), shortTone );
delay ( shortTone );
```

```
tone ( BZ, scale2Hz ( "mi" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "fa" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "so" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "ra" ), longTone );
delay ( longTone + 150 );

tone ( BZ, scale2Hz ( "ra" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "re" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "mi" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "fa" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "so" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "ra" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "si" ), longTone );
delay ( longTone + 150 );

tone ( BZ, scale2Hz ( "si" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "mi" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "fa" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "so" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "ra" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "si" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "do" ), longTone );
delay ( longTone + 150 );

tone ( BZ, scale2Hz ( "do" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "si" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "ra" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "fa" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "si" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "so" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "do" ), longTone );
delay ( longTone + 150);
```

```
tone ( BZ, scale2Hz ( "do" ), veryShortTone );
delay ( veryShortTone );
tone ( BZ, scale2Hz ( "mi" ), veryShortTone );
delay ( veryShortTone + 50 );
tone ( BZ, scale2Hz ( "mi" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "mi" ), veryShortTone );
delay ( veryShortTone );
tone ( BZ, scale2Hz ( "so" ), veryShortTone );
delay ( veryShortTone + 50 );
tone ( BZ, scale2Hz ( "so" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "re" ), veryShortTone );
delay ( veryShortTone );
tone ( BZ, scale2Hz ( "fa" ), veryShortTone );
delay ( veryShortTone + 50 );
tone ( BZ, scale2Hz ( "fa" ), shortTone );
delay ( shortTone );
tone ( BZ, scale2Hz ( "ra" ), veryShortTone );
delay ( veryShortTone );
tone ( BZ, scale2Hz ( "si" ), veryShortTone );
delay ( veryShortTone + 50);
tone ( BZ, scale2Hz ( "si" ), shortTone );
delay ( shortTone + 100 );

tone ( BZ, scale2Hz ( "so" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "do" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "ra" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "fa" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "mi" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "do" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "re" ), longTone );
delay ( longTone + 200);

tone ( BZ, scale2Hz ( "so" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "do" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "ra" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "si" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "do" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "re" ), longTone );
delay ( longTone );
tone ( BZ, scale2Hz ( "do" ), veryLongTone );
delay ( veryLongTone );
```

```
}

/**
 * 関数名: scale2Hz(自作関数)
 * 引数: scale: 音階
 * 処理: 音階から周波数を計算する
 * 返り値: 周波数[Hz]
 */
float scale2Hz ( String scale ) {
    float f = 0.0;              // 周波数[Hz]
    float baseHz = 440.0;       // 基準周波数("ラ"の周波数)[Hz]
    float octaveKeyNum = 12.0;  // 1オクターブの音階数

    // 音階から周波数を決定
    if ( scale == "fam1" || scale == "fm1" ) {
        f = baseHz * pow ( 2.0, ( -16.0 / octaveKeyNum ) );
    } else if ( scale == "fa#m1" || scale == "f#m1" ) {
        f = baseHz * pow ( 2.0, -15.0 / octaveKeyNum );
    } else if ( scale == "som1" || scale == "gm1" ) {
        f = baseHz * pow ( 2.0, -14.0 / octaveKeyNum );
    } else if ( scale == "so#m1" || scale == "g#m1" ) {
        f = baseHz * pow ( 2.0, -13.0 / octaveKeyNum );
    } else if ( scale == "ram1" || scale == "am1" ) {
        f = baseHz * pow ( 2.0, -12.0 / octaveKeyNum );
    } else if ( scale == "ra#m1" || scale == "a#m1" ) {
        f = baseHz * pow ( 2.0, -11.0 / octaveKeyNum );
    } else if ( scale == "sim1" || scale == "bm1" ) {
        f = baseHz * pow ( 2.0, -10.0 / octaveKeyNum );
    } else if ( scale == "do" || scale == "c" ) {
        f = baseHz * pow ( 2.0, -9.0 / octaveKeyNum );
    } else if ( scale == "do#" || scale == "c#" ) {
        f = baseHz * pow ( 2.0, -8.0 / octaveKeyNum );
    } else if ( scale == "re" || scale == "d" ) {
        f = baseHz * pow ( 2.0, -7.0 / octaveKeyNum );
    } else if ( scale == "re#" || scale == "d#" ) {
        f = baseHz * pow ( 2.0, -6.0 / octaveKeyNum );
    } else if ( scale == "mi" || scale == "e#" ) {
        f = baseHz * pow ( 2.0, -5.0 / octaveKeyNum );
    } else if ( scale == "fa" || scale == "f" ) {
        f = baseHz * pow ( 2.0, -4.0 / octaveKeyNum );
    } else if ( scale == "fa#" || scale == "f#" ) {
        f = baseHz * pow ( 2.0, -3.0 / octaveKeyNum );
    } else if ( scale == "so" || scale == "g" ) {
        f = baseHz * pow ( 2.0, -2.0 / octaveKeyNum );
    } else if ( scale == "so#" || scale == "g#" ) {
        f = baseHz * pow ( 2.0, -1.0 / octaveKeyNum );
    } else if ( scale == "ra" || scale == "a" ) {
        f = baseHz * pow ( 2.0, 0.0 / octaveKeyNum );
    } else if ( scale == "ra#" || scale == "a#" ) {
        f = baseHz * pow ( 2.0, 1.0 / octaveKeyNum );
    } else if ( scale == "si" || scale == "b" ) {
        f = baseHz * pow ( 2.0, 2.0 / octaveKeyNum );
```

```
    } else if ( scale == "dop1" || scale == "cp1" ) {
        f = baseHz * pow ( 2.0, 3.0 / octaveKeyNum );
    } else if ( scale == "do#p1" || scale == "c#p1" ) {
        f = baseHz * pow ( 2.0, 4.0 / octaveKeyNum );
    } else if ( scale == "rep1" || scale == "dp1" ) {
        f = baseHz * pow ( 2.0, 5.0 / octaveKeyNum );
    } else if ( scale == "re#p1" || scale == "d#p1" ) {
        f = baseHz * pow ( 2.0, 6.0 / octaveKeyNum );
    } else if ( scale == "mip1" || scale == "ep1" ) {
        f = baseHz * pow ( 2.0, 7.0 / octaveKeyNum );
    } else if ( scale == "fap1" || scale == "fp1" ) {
        f = baseHz * pow ( 2.0, 8.0 / octaveKeyNum );
    } else if ( scale == "fa#p1" || scale == "f#p1" ) {
        f = baseHz * pow ( 2.0, 9.0 / octaveKeyNum );
    } else if ( scale == "sop1" || scale == "gp1" ) {
        f = baseHz * pow ( 2.0, 10.0 / octaveKeyNum );
    } else if ( scale == "so#p1" || scale == "g#p1" ) {
        f = baseHz * pow ( 2.0, 11.0 / octaveKeyNum );
    } else if ( scale == "rap1" || scale == "ap1" ) {
        f = baseHz * pow ( 2.0, 12.0 / octaveKeyNum );
    } else if ( scale == "ra#p1" || scale == "a#p1" ) {
        f = baseHz * pow ( 2.0, 13.0 / octaveKeyNum );
    } else if ( scale == "sip1" || scale == "bp1" ) {
        f = baseHz * pow ( 2.0, 14.0 / octaveKeyNum );
    } else if ( scale == "dop2" || scale == "cp2" ) {
        f = baseHz * pow ( 2.0, 15.0 / octaveKeyNum );
    }

    return f;  // 周波数を返す[Hz]
}
```