```
Script started on Tue 15 May 2012 11:09:58 PM CDT
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ p
wd
/home/georgia/cplusplus
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ c
at counterpoint.cpp
#include <iostream>
#include <vector>
#include <cmath>
#include <climits>
#include <string>
#include <ctime>
#include <cstdlib>

using namespace std;

/* String representations of all the notes in two octaves. Throughout western
music */

const string allNotes[] = {"C,", "C#,","Db,", "D,", "D#,", "Eb,", "E,", "E#,",
                           "F,", "F#,", "Gb,", "G,", "G#,", "Ab,", "A,", "A#,",
                           "Bb,", "B,", "B#,", "C", "C#", "Db", "D", "D#", "Eb",
                           "E", "E#", "F", "F#", "Gb", "G", "G#", "Ab", "A", "A#",
                           "Bb", "B", "B#", "C'"};

const short modec[] = {0, 3, 6, 8, 11, 14, 17, 19, 22, 25, 27, 30, 33, 36, 38};

const short cons[] = {0, 5, -5, 6, -6, 11, -11, 13, -13, 14, -14, 19, 24, 25};

const short diss[] = {1, 4, 7, 9, 10, 12, 15, 16, 17, 18};

const short perfect[] = {0, 11, 19};




inline vector<short>::size_type rand_vec(vector<short> vec)
{
    vector<short>::size_type max = vec.size();
    return rand() % max;
}


inline short harm(short pitch1, short pitch2)
{
    return abs(pitch2 - pitch1);
}

inline short mel(short note, short prevNote)
{
    return note - prevNote;
}

void read(vector<short> & vec);

void print(vector<short> vec);

bool is_perfect(short intvl);

bool is_parallel(short intvl1, short intvl2);

bool is_direct(short intvl, short upper_int, short lower_int);

bool motion_ok(short intvl, short prevInt);

bool motion_ok(short intvl);

bool in_mode(short note);

short repeat(vector<short> vec, vector<short>::size_type start,
            vector<short>::size_type end);




int main(void)
{
    srand(time(NULL));
    vector<short>::size_type i;
    vector<short>::size_type c;
    short k, option;
    vector<short> options;
    vector<short> cantus;
    vector<short> counterpoint;


    cout << "\nEnter the Cantus pitches in order ('q' to end):\n";
    read(cantus);



    k = 0;

    /* Begin overall loop which provides us with as many solutions as can be
    completed in 50 cycles. */

    while (k != 100)
    {
        c = 1;

        counterpoint.clear();

        /* The first note of the counterpoint is always a unison, fifth, or
        octave, so we add the first note without checking anything. */

        counterpoint.push_back(cantus[0] + 19);

        /* loops through each note of the cantus unless it reaches a point where
        there are no longer any legal possibilities available*/
        do
        {
            options.clear();

            /*loop through each of the consonant intervals stored in the cons
            array*/

            for (i = 0; i < 14; i++) // consonance loop
            {
                /* assign the sum of the cantus pitch and the consonant interval
                to the option variable for readablity */

                option = cantus[c] + cons[i];

                /* For the second pitch of the counterpoint, test for parallel
                or direct motion, and test the conotour with the single-argument
                version which only looks back one note. Check that the pitch is
                in the C mode, and that it doesn't repeat the previous pitch. */

                if (c > 1 && !is_parallel(cons[i],
                               harm(cantus[c-1],  counterpoint[c-1]) ) &&
                    !is_direct(cons[i],
                        mel(option,  counterpoint[c-1]),
                        mel(cantus[c], cantus[c-1])) &&


                    motion_ok(mel(option,  counterpoint[c-1]),
```

```
                    mel( counterpoint[c-1],  counterpoint[c-2])) &&

                in_mode(option) && (option !=  counterpoint[c-1]))
            {
                options.push_back(option);
            }

            /* same test as above, but after pitch 2 use the motion_ok
            function with two arguments to test for too much disjunct
            motion*/

            if (c <= 1 && !is_parallel(cons[i],
                              harm(cantus[c-1],  counterpoint[c-1]) ) &&
                !is_direct(cons[i],
                    mel(option,  counterpoint[c-1]),
                    mel(cantus[c], cantus[c-1])) &&


                motion_ok(mel(option,  counterpoint[c-1])) &&

                in_mode(option) && (option !=  counterpoint[c-1]))
            {
                options.push_back(option);
            }
        }

        /*if there are any legal options, add one randomly chosen option to
        the counterpoint vector and continue on to the next cantus note. */

        if (options.size() > 0)
        {
            counterpoint.push_back(options[rand_vec(options)]);
        }

        c++;


    } while (c < cantus.size()-1 && options.size() != 0); // end cantus loop

    option = cantus[c] + 19;

    if (options.size() != 0 && !is_parallel(19, harm(cantus[c-1],
                                                counterpoint[c-1]) ) &&
                !is_direct(19, mel(option,  counterpoint[c-1]),
                                            mel(cantus[c], cantus[c-1])) &&
                            motion_ok(mel(option,  counterpoint[c-1])) &&
                        in_mode(option) && (option !=  counterpoint[c-1]))
    {
         counterpoint.push_back(option);
    }

    /* Check for the frequency of appearance of each pitch in the
    counterpoint. Depending on the length of the cantus this would be
    higher or lower, but for 8 -10 pitches, the melody sounds aimless if
    any single pitch is repeated more than twice. */

    if (repeat(counterpoint, 1, 1) > 1)
    {
        counterpoint.clear();
    }


    if (counterpoint.size() == cantus.size())
    {
        cout << "\nCounterpoint number " << k << ": ";
        print(counterpoint);
    }
    k++;
}
```

```
    cout << "\nCantus: ";
    print(cantus);


    return 0;

}

bool is_perfect(short intvl)
{
    short i = 0;
    bool is;

    while (i < 3 && intvl != perfect[i])
    {
        i++;
    }
    if (i == 3)
    {
        is = false;
    }
    else
    {
        is = true;
    }

    return is;
}
bool is_parallel(short intvl1, short intvl2)
{
    return (is_perfect(intvl1) && is_perfect(intvl2));
}

void read(vector<short> & vec)
{
    short pos;
    string note;

        cin >> note;

    while (toupper(note[0]) != 'Q')
    {
        pos = 0;

        while (pos != 39 && note != allNotes[pos])
        {
            pos++;
        }
        vec.push_back(pos);
        cin >> note;
    }
    cin.clear();
    cin.ignore(INT_MAX, '\n');

    return;
}

void print(vector<short> vec)
{
    vector<short>::size_type pos;

    for( pos = 0; pos < vec.size() - 1; pos++)
    {
        cout << allNotes[vec[pos]] << "  ";
    }

    cout << allNotes[vec[vec.size() - 1]] << "\n";
```

```
    return;
}

bool is_direct(short intvl, short upper_int, short lower_int)
{
    bool isdir;

    if (is_perfect(intvl) && ((upper_int > 0 && lower_int > 0) ||
                              (upper_int < 0 && lower_int < 0 )))
    {
        isdir = true;
    }
    else
    {
        isdir = false;
    }

    return isdir;
}


bool motion_ok(short intvl, short prevInt)
{
    short i = 0;

    while ((i != 10) &&
           (abs(intvl) != diss[i]) &&
       /* if previous motion was a leap up or down the voice cannot continue in
       the same direction, and must move by step*/
             !(prevInt > 6 && intvl > 0) &&
             !(prevInt < -6 && intvl < 0) &&
             !(abs(prevInt) > 3 && abs(intvl) > 3)  &&
           (abs(intvl) < 11 /*|| abs(intvl) == 19*/))
    {
        i++;
    }

    return i == 10;
}
bool motion_ok(short intvl)
{
    short i = 0;

    while ((i != 10) &&
           (abs(intvl) != diss[i]) &&
           (abs(intvl) < 11 /*|| abs(intvl) == 19*/))
    {
        i++;
    }

    return i == 10;
}

bool in_mode(short note)
{
    short i = 0;
    while (i != 15 && note != modec[i])
    {
        i++;
    }

    return i != 15;
}

short repeat(vector<short> vec, vector<short>::size_type start,
             vector<short>::size_type end)
{
    short i = 0;
    vector<short>::size_type c;
    vector<short>::size_type j;

    c = start;

    while (c != vec.size() - end && i < 2)
    {
        j = 1;
        i = 0;
        while ( j != vec.size() - end - start)
        {
            if (vec[c] == vec[(c+j) % (vec.size() - end - start)])
            {
                i++;
            }
            j++;
        }
        c++;
    }

    return i;
}



\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ C
PP count\007erpoint\033[K
counterpoint.cpp***
counterpoint.cpp: In function â\200\230short int harm(short int, short
int)â\200\231:
counterpoint.cpp:41:31: warning: conversion to â\200\230short intâ\200\231 from
â\200\230intâ\200\231 may alter its value [-Wconversion]
counterpoint.cpp: In function â\200\230short int mel(short int, short
int)â\200\231:
counterpoint.cpp:46:19: warning: conversion to â\200\230short intâ\200\231 from
â\200\230intâ\200\231 may alter its value [-Wconversion]
counterpoint.cpp: In function â\200\230int main()â\200\231:
counterpoint.cpp:102:46: warning: conversion to â\200\230std::vector<short
int>::value_type {aka short int}â\200\231 from â\200\230intâ\200\231 may alter its
value
[-Wconversion]
counterpoint.cpp:118:44: warning: conversion to â\200\230short intâ\200\231 from
â\200\230intâ\200\231 may alter its value [-Wconversion]
counterpoint.cpp:172:30: warning: conversion to â\200\230short intâ\200\231 from
â\200\230intâ\200\231 may alter its value [-Wconversion]
counterpoint.cpp: In function â\200\230bool motion_ok(short int, short
int)â\200\231:
counterpoint.cpp:297:33: warning: comparing floating point with ==
or != is unsafe [-Wfloat-equal]
counterpoint.cpp: In function â\200\230bool motion_ok(short int)â\200\231:
counterpoint.cpp:315:33: warning: comparing floating point with ==
or != is unsafe [-Wfloat-equal]


\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ .
/counterpoint.out

Enter the Cantus pitches in order (’q’ to end):
D, A, G, F, E, D, F, E, D, q

Counterpoint number 8: D  F  G  A  G  F  D  C  D

Counterpoint number 20: D  C  B,  A,  G,  B,  A,  C  D

Counterpoint number 29: D  F  E  A  G  F  D  C  D

Counterpoint number 40: D  C  B,  A,  G,  A,  D  C  D

Counterpoint number 66: D  F  E  A  G  F  D  C  D
```

```
Counterpoint number 85: D  C  E  F  G  F  D  C  D

Counterpoint number 86: D  C  E  F  G  F  D  C  D

Cantus: D,  A,  G,  F,  E,  D,  F,  E,  D,
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ .
/counterpoint.out

Enter the Cantus pitches in order ('q' to end):
C, D, E, F, G♭ E, F, B♭ C, q

Counterpoint number 0: C  F  E  D  E  C  D  B,  C

Counterpoint number 6: C  F  E  D  B,  C  D  B,  C

Counterpoint number 8: C  F  E  D  B,  C  A,  B,  C

Counterpoint number 13: C  F  E  D  E  C  D  B,  C

Counterpoint number 21: C  F  E  D  E  C  D  B,  C

Counterpoint number 24: C  F  E  D  E  C  D  B,  C

Counterpoint number 31: C  F  E  D  B,  C  A,  B,  C

Counterpoint number 38: C  F  E  D  E  C  D  B,  C

Counterpoint number 46: C  F  E  D  B,  C  A,  B,  C

Counterpoint number 47: C  B,  G,  F,  E,  G,  A,  B,  C

Counterpoint number 55: C  F  E  D  E  C  D  B,  C

Counterpoint number 65: C  F  E  D  B,  C  A,  B,  C

Counterpoint number 76: C  F  E  D  E  C  D  B,  C

Counterpoint number 77: C  F  E  D  B,  C  A,  B,  C

Counterpoint number 83: C  F  E  D  B,  C  A,  B,  C

Counterpoint number 89: C  B,  G,  F,  E,  G,  A,  B,  C

Counterpoint number 91: C  F  E  D  E  C  D  B,  C

Cantus: C,  D,  E,  F,  G,  E,  F,  D,  C,
\033]0;georgia@georgia-MT6017: ~/cplusplus\007georgia@georgia-MT6017:~/cplusplus$ e
xit
exit

Script done on Tue 15 May 2012 11:11:44 PM CDT
```