The Pairs

- Lilly Webster and Tyler Williams
- Zachary Susag and Tanner Tufto
- Cory McCartan and Anna Blindermann
- Dennis Chan and Eli Salm
- Josh Lavin and Lex Martin
- Faizaan Ali and Colin Greenman
- Mattori Birnbaum and Erhaan Ahmad
- Sanjay Sudhir and Jae Hong Shin
- Sophie Gaschott and Yuyin Sun
- Jong Hoon Bae and Ella Nicolson
- Ying Zhang and Gemma Nash

Saung Thuya will replace anyone who is absent on Wednesday.

The Assignment

The project is to create and play a musical composition for the Scribbler2 robot’s sound card in the form of a theme with variations.

The theme will consist of fourteen notes altogether. Your program should choose the first six at random from the “just temperament” musical scale presented in two of the previous handouts (most recently in the lab on header files and inclusion). The seventh note should be the “sol” note on that scale. The next six notes should once more be chosen at random, and the last note should be the “do” at the bottom of the scale. In the theme, the twelve randomly selected notes should be equal in duration, and the two fixed notes (the seventh and fourteenth) should be twice as long.

Each of the variations should be the result of applying some systematic transformation to the theme. The variation should sound different from the theme in some way but should also be discernibly related to the theme, although the nature of the transformation need not be obvious to the listener.

Your program should construct the theme and then have the Scribbler2 robot play the theme, followed by each of four or more variations, separated by pauses of five seconds.

To choose a random note from the eight-note just-temperament scale, you’ll need to set up the scale in an array (as in the lab on header files) and then generate a random number in the range from 0 up to, but not including, 8 to use as a subscript when indexing into this array.

Here is a fairly straightforward way to obtain random numbers, using functions and constants from the standard library whose header file is stdlib.h:

- Initialize the pseudo-random-number generator (PRNG) that is provided as part of the run-time system by invoking the srand function, which takes an unsigned int as its argument and uses that value as the “initial state” of the PRNG. If you always provide the same unsigned int to srand, the generator will always produce the same sequence of random numbers. If you prefer to have different random numbers on each run of the program, use the statement

  srand((unsigned) time(NULL));
which examines the computer’s internal clock and uses the time reading that it returns as the initial state of the PRNG. (To use time function, you’ll need to include the time.h header file.)

• The rand function, which takes no arguments, will now return a random number in the range from 0 up to and including RAND_MAX (which on our workstations is defined to be 2147483647, the same as INT_MAX). To convert this into a number in the range from 0 up to (but not including) ARRAY_SIZE we can rescale it, taking care to avoid integer overflows.

random_subscript = (rand() * (long) ARRAY_SIZE) / (long) RAND_MAX;

Note, however, that this accommodation works on our workstations only because the long type has sixty-four bits, enough to hold the products that we’re going to be computing. For greater portability, one could write

random_subscript = (int) (ARRAY_SIZE * (rand() / (double) RAND_MAX));

 Submitting the Program

I’ll collect the programs in hard copy at the beginning of class on Monday, February 29.

We’re still working on a mechanism for submitting programs electronically without sharing them to the world. Until that’s ready, just include the full pathname of your program in the opening comment, and I’ll pull a copy directly from inside your home directory.