Using the supplied code

There are two files of supplied code you will need. The n2.o file and the libmidi.a file provide needed functions. They both have header files, n2.h and midi.h that some of your code will have to include.

Library Function list

```
int midi_X(int color, double X, double Y);
void midi_beep();
void midi_clear();
void midi_flash();
int midi_getch();
int midi_initialize(int debug);
int midi_projectile(int color, double X, double Y, double direction);
void midi_refresh();
int midi_smile(int score, int color, double X, double Y, double
direction);
void midi_status(const char *statstr);
void midi_teardown();
void midi_time(int milliseconds);
bool midi_touches_wall(double X, double Y);
```


The initialization code will need to call midi_initialize and pay close attention to the return value. The clear up code after everything is over calls midi_teardown. The sim needs midi_touches_wall and all of the others are for graphical output.

The functions in n2


```
int microsleep(unsigned int microseconds);
int millisleep(unsigned int milliseconds);
double now();
```

The n2.o file contains the code for the functions that deal with time. The main function uses now and the graphical output needs to call microsleep.

Sample Makefile fragment

Here is how the reference code builds:

```
lab2: lab2.o n2.o input.o bits.o output.o sim.o
        gcc -g -o $@ $^  -lm -L. -lmidi -lncurses
```

The executable depends on .o files and dot o file are created by the master rule you already have. The executable needs the math library, so it add **-lm** to the command line. The midi maze library is not in the standard place for libraries, so **-L.** is required to tell gcc to search in dot ".", which is the current directory, to find libraries. The **-lmidi** tells gcc to build with the midi maze library, which needs to be in your lab2 directory. The library needs the new curses library, so we have **-lncurses** as well.

The names of the other .o files will need to match your code.