```
#include <stdio.h>

static int              green = 0;
static int              red = 1;

static double           MHR[ 11 ] = { 0 };
static size_t           MHRSZ = sizeof( MHR ) / sizeof( MHR[ 0 ] );

static void             initMHR();
static int              hasMHR( double, double );

extern int
main( int argc, char *argv[] )
{
      int               state;
      int               mhrndx = -1;
      int               greenN = 0;
      int               redN = 0;
      double            elapsedtm = 0.0;
      double            prevbtm = 0.0;
      double            nextbtm = 0.0;
      double            greentm = 0.0;
      double            redtm = 0.0;
      double            partialdelta = 0.0;
      double            remainingdelta = 0.0;
      double            fulldelta = 43200.0 / 1427.0;

      initMHR();

      state = green;
      nextbtm = fulldelta;

      while ( ( int )elapsedtm < 43200 )
      {
            mhrndx = hasMHR( prevbtm, nextbtm );
            if ( mhrndx < 0 )
            {
                  if ( state == green )
                  {
                        greentm += fulldelta;
                        greenN++;
                  }
                  else
                  {
                        redtm += fulldelta;
                        redN++;
                  }
            }
            else
            {
                  partialdelta = MHR[ mhrndx ] - prevbtm;
                  remainingdelta = fulldelta - partialdelta;

                  if ( state == green )
```

```
                {
                        greentm += partialdelta;
                        greenN++;
                        redtm += remainingdelta;
                        redN++;
                }
                else
                {
                        redtm += partialdelta;
                        redN++;
                        greentm += remainingdelta;
                        greenN++;
                }

                state = state == green ? red : green;
        }
        elapsedtm += fulldelta;
        prevbtm = nextbtm;
        nextbtm += fulldelta;
        state = state == green ? red : green;
    }
    greentm -= ( elapsedtm - 43200.0 );
    state = state == green ? red : green;

    printf( "\nfulldelta = %12.9f\n"
        "green tm = %f N = %d\n"
        "red tm = %f N = %d\n"
        "green tm + red tm = %f\n"
        "green tm - red tm = %11.9f\n"
        "last state was %s\n"
        "P(green) = %11.9f\n"
        "P(red) = %11.9f\n",
            fulldelta,
            greentm, greenN,
            redtm, redN,
            greentm + redtm,
            greentm - redtm,
            state == green ? "green" : "red",
            greentm / 43200.0,
            redtm / 43200.0 );

    return 0;
}

static void
initMHR()
{
    int             n;
    int             hrs;
    int             mins;
    double          secs;

    for ( n = 1; n <= MHRSZ; n++ )
    {
```

```
            hrs = n;
            mins = ( n * 60 ) / 11;
            secs = ( ( n * 60.0 ) / 11.0 - mins ) * 60;
            MHR[ n - 1 ] = 3600 * hrs + 60 * mins + secs;
      }

      printf( "MHR:\n" );
      for ( n = 0; n < MHRSZ; n++ )
      {
            hrs = MHR[ n ] / 3600;
            mins = ( MHR[ n ] - hrs * 3600 ) / 60;
            secs = MHR[ n ] - hrs * 3600 - mins * 60;
            printf( "%02d:%02d:%012.9f\n", hrs, mins, secs );
      }
}

static int
hasMHR( double prevbtm, double nextbtm )
{
      int               n;

      for ( n = 0; n < ( MHRSZ - 1 ); n++ )
      {
            if ( MHR[ n ] > prevbtm && MHR[ n ] < nextbtm )
            {
                  return n;
            }
      }

      return -1;
}
```