```
1.
**************gpio.h**************

#ifndef GPIO_H
#define GPIO_H

/* takes each segment's 10 time indices and blinks LEDs appropiatly for
 * 10-second segment
 */
void blink(char *segment);

/* takes parsed year, month, and day; returns calculated 'day of year' */
int calcdoy(char *year, char *month, char *day);

/* each 'seg' func returns 10 bit string to be passed to 'blink()' func */
char *segzero(char *segbuff);
char *segfive(char *segbuff, int doy);
char *segfour(char *segbuff, int doy);
char *segthree(char *segbuff, char *times);
char *segtwo(char *segbuff, char *times);
char *segone(char *segbuff, char *year, int dstflag);

/* takes the parsed year field as an integer; returns 1 if leap year, returns * 0 if not
 */
int isleapyear(int year);

/* takes an integer and returns 4 bit BCD value */
char *int2bin(int dec);

/* gpio functions to set up GPIO#12 */
void exportgpio(char *pin);
void unexportgpio(char *pin);
void setvalue(char *pin, char *onezero);
void pindirection(char *pin, char *inout);

#endif



**************wwv.c**************

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include "gpio.h"

void blink(char *segment)
{
    /* initialize input strings to blink LED on GPIO#12 */
    char pin[3] = "12";
    char one[2] = "1";
    char zero[2] = "0";
    int i = 0;

    for (i = 0; i < 10; i++) {
        /* zero bit(0) represented by LED on for 170ms, off for rest
         * of second
            */
        if (segment[i] == 0) {
            setvalue(pin, one);
            usleep(170000);
            setvalue(pin, zero);
            usleep(830000);
        /* one bit(1) represented by LED on for 470ms, off for rest
         * of second
         */
        } else if (segment[i] == 1) {
            setvalue(pin, one);
            usleep(470000);
            setvalue(pin, zero);
            usleep(530000);
        /* position identifier(2) represented by LED on for 770ms,
         * off for rest of second
             */
        }else if ( segment[i] == 2) {
            setvalue(pin, one);
            usleep(770000);
            setvalue(pin, zero);
            usleep(230000);
        /* always zero(3) represented by LED off for duration of
             * one second
             */
        } else if ( segment[i] == 3) {
            setvalue(pin, zero);
            usleep(1000000);
        }
    }
}

int calcdoy(char *year, char *month, char *day)
{
    int yearint, isleap, doy, flag, dayint;
    int days[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    char *months[12] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct
", "Nov", "Dec"};

    /* change days in February based on leap year */
    yearint = atoi(year);
    isleap = isleapyear(yearint);

    if (isleap) {
        days[1] = 29;
    }

    dayint = atoi(day);
    /* initialize day of year with number of days in current month */
    doy = dayint;
    flag = 0;
    for (int i = 11; i >= 0; --i) {
        if (flag) {
            doy += days[i];
        }
        if (strcmp(month, months[i]) == 0) {
            flag = 1;
        }
    }
return doy;
}

char *segzero(char *segbuff)
{
    int i;

    /* fill buffer with ’always zero’ (3) for last 10 seconds of wwv
     * pattern
     */
    for (i = 0; i < 10; i++) {
        segbuff[i] = 3;
    }
return segbuff;
}

char *segfive(char *segbuff, int doy)
```

```
{
    char buff[3];
    int doyhund, i;
    char *bin;

    /* convert day of year to string to be parsed by ones, tens, and
     * hundreds place.
     */
    snprintf(buff, 10, "%d", doy);
    if (strlen(buff) == 2) {
        doyhund = 0;
    }
    else {
        doyhund = buff[0];
    }

    /* pass day of year hundred’s place to helper function to return
     * BCD.
     */
    bin = int2bin(doyhund);
    for (i = 0; i < 4; i++) {
        segbuff[i] = bin[i];
    }
    free(bin);

    segbuff[4] = 0;
    for (i = 0; i < 5; i++) {
        segbuff[i + 5] = 3;
    }
return segbuff;
}

char *segfour(char *segbuff, int doy)
{
    char buff[3];
    int doyones, doytens, i;
    char *bin;

    snprintf(buff, 10, "%d", doy);
    if (strlen(buff) == 2) {
        doyones = buff[1];
        doytens = buff[0];
    }
    else {
        doyones = buff[2];
        doytens = buff[1];
    }

    bin = int2bin(doyones);
    for (i = 0; i < 4; i++) {
        segbuff[i] = bin[i];
    }
    free(bin);
    segbuff[4] = 0;

    bin = int2bin(doytens);
    for (i = 0; i < 4; i++) {
        segbuff[i + 5] = bin[i];
    }
    free(bin);

    /* pos identifier */
    segbuff[9] = 2;
return segbuff;
}

char *segthree(char *segbuff, char *times)
{
    int i, hr_ones, ho, hr_tens, ht;
    char *bin;

    /* extract one’s place of hours */
    hr_ones = times[1];
    ho = hr_ones - ’0’;

    bin = int2bin(ho);
    for (int i = 0; i < 4; i++) {
        segbuff[i] = bin[i];
    }
    free(bin);

    segbuff[4] = 0;

    /* extract tens place of hours */
    hr_tens = times[0];
    ht = hr_tens - ’0’;

    bin = int2bin(ht);
    for (i = 0; i < 4; i++) {
        segbuff[i + 5] = bin[i];
    }
    free(bin);

    /* pos identifier */
    segbuff[9] = 2;

return segbuff;
}

char *segtwo(char *segbuff, char *times)
{
    int i, min_ones, mo, men_tens, mt;
    char *bin;

    /* extract one’s place of minutes */
    min_ones = times[4];
    mo = min_ones - ’0’;
    bin = int2bin(mo);
    for (i = 0; i < 4; i++) {
        segbuff[i] = bin[i];
    }
    free(bin);

    segbuff[4] = 0;

    /* extract ten’s place */
    men_tens = times[3];
    mt = men_tens - ’0’;

    bin = int2bin(mt);
    for (i = 0; i < 4; i++) {
        segbuff[i + 5] = bin[i];
    }
    free(bin);

    /* pos identifier */
    segbuff[9] = 2;

return segbuff;
}

char *segone(char *segbuff, char *year, int dstflag)
{
    int i, yearint, y;
    char *bin;
    segbuff[0] = 3;
    segbuff[1] = 0;
    segbuff[2] = dstflag;

    /* check if leapyear using helper function */
    yearint = atoi(year);
    segbuff[3] = isleapyear(yearint);

    y = year[3] - '0';

    bin = int2bin(y);
    for (i = 0; i < 4; i++) {
        segbuff[i + 4] = bin[i];
    }
    segbuff[8] = 0;
    segbuff[9] = 2;
    free(bin);
return segbuff;
}

int isleapyear(int year)
{
    if (~(year % 4)) {
        if (~(year % 100)) {
            if (~(year % 400)) {
                return 1;
            } else {
                return 0;
            }
        } else {
            return 1;
        }
    } else {
        return 0;
    }
}

char *int2bin(int a)
{
    char *bin;
    /* allocate four bytes for BCD value to be returned */
    bin = (char *)malloc(4 * sizeof(char));

    (a & 0x1) ? (bin[0] = 1) : (bin[0] = 0);
    (a & 0x2) ? (bin[1] = 1) : (bin[1] = 0);
    (a & 0x4) ? (bin[2] = 1) : (bin[2] = 0);
    (a & 0x8) ? (bin[3] = 1) : (bin[3] = 0);
return bin;
}



***************main.c***************

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
#include "gpio.h"

int main(int argc, char *argv[])
{
    struct tm *datetime;
    time_t t;
    char *utc, *tok, *arr[5], *year, *times;
    char *seg1, *seg2, *seg3, *seg4, *seg5, *seg0;
    int i, dstflag;
    char *segbuff;
    char pin[3] = "12";
    char inout[4] = "out";

    /* allocate memory to hold a 'segment buffer', holding each 10 time
     * indexes upon iteration.
     */
    segbuff = (char *)malloc(10 * sizeof(int));


    /* create "gpio12" node */
    exportgpio(pin);
    usleep(1000000);
    /* write 'out' to GPIO#12 setting it as an output */
    pindirection(pin, inout);
    usleep(1000000);
    printf("Begin wwv signal.\n");

    /* generate current time and date in UTC */
    t = time(NULL);
    datetime = gmtime(&t);
    utc = asctime(datetime);
    /* read 'tm_isdst' flag based off current date to check if in DST */
    dstflag = datetime->tm_isdst;

    /* UTC time and dat in format:
     * 'Thu Mar 5 17:56:40 2020'.
     *  parse into tokens and store into 'arr[]'.
     */
    tok = strtok(utc, " ");
    i = 0;
    while (tok != NULL) {
        arr[i++] = tok;
        tok = strtok(NULL, " ");
    }

    /* 'Segment 1' blinking */
    year = arr[4];
    seg1 = segone(segbuff, year, dstflag);
    blink(seg1);

    /* 'Segment 2' blinking */
    times = arr[3];
    seg2 = segtwo(segbuff, times);
    blink(seg2);

    /* 'Segment 3' blinking */
    seg3 = segthree(segbuff, times);
    blink(seg3);

    /* 'Segment 4' blinking */
    char *month = arr[1];
    char *day = arr[2];
    /* calculate day of year with helper function */
    int doy = calcdoy(year, month, day);
    seg4 = segfour(segbuff, doy);
    blink(seg4);

    /* 'Segment 5' blinking */
    seg5 = segfive(segbuff, doy);
    blink(seg5);

    /* 'Segment 0' blinking */
    seg0 = segzero(segbuff);
    blink(seg0);

    /* free allocated memory */
    free(segbuff);
    /* remove "gpio12" node */
    printf("End of signal.\n");
    unexportgpio(pin);
    usleep(1000000);

return 0;
}
```

```
***************Makefile***************

TARGET=gpioLED
OBJS=main.o gpio.o wwv.o
CFLAGS=-g -Wall -O2

all: ${TARGET}

${TARGET}: ${OBJS}
    ${CC} -o ${TARGET} ${OBJS}

clean:
    rm -f ${TARGET} ${OBJS} core*
gpio.o: gpio.h
wwv.o: gpio.h
```

2.
a)

```
    sudo apt-get update
    sudo apt-get upgrade
    sudo reboot
```

b)

```
    cd /usr/src
    sudo get-apt install bison bc flex libssl-dev build-essentials
    sudo chown -R pi /usr/src
    git clone git@github.com:raspberrypi/linux.git
```

c)

```
    uname -r
```

d)

```
    mv linux linux4.19.97
```

e)

```
    make mrproper
    sudo modprobe configs
    cat /proc/config.gz | gunzip > .config
    wget https://github.com/raspberrypi/firmware/raw/-1ecfd2ba2b7cf3a2f4aa75ada895ee4a3e729
f5/extra/Module7l.symvers
    sudo mv Module7l.symvers /boot
    ln -s /boot/Module7l.symvers Module.symvers
    make modules-prepare
```

f)

```
    cd /lib/modules/4.19.97-v7+
    sudo ln -s /usr/src/linux4.19.97-v7+ build
```

g)

```
    sudo ln -s build source
```

*DO NOT sudo apt-get update or upgrade from this point on

3.
a)

```
***************perlgps***************

#!/usr/bin/perl

# forces to initialize local vars with 'my' keyword
use strict;
# helps find typing mistakes
use warnings;

# try to open file (first CLA) with 'FILE' handler
open (FILE, $ARGV[0]) or die "Cannot open file\n";

# execute until eof
while (<FILE>) {
    # remove newline charact '$_'
    chomp;
    # split lines by commas, store into array '@fields'
    my @fields = split(',');
    # print specified field (column) of file
    print "$fields[$ARGV[1]]\n";
}
close(FILE);
```

b)

```
***************perlstat***************

#!/usr/bin/perl

# forces to initialize local vars with 'my' keyword
use strict;
# helps find typing mistakes
use warnings;

# iterate through each CLA filename
foreach my $filename (@ARGV) {
    # extract number of hard links through stat() func
    my $link = (stat($filename))[3];
    # print # of hard links and file name if and only if
    # number of hard links > 1
    print "$link $filename\n" if ($link > 1);
}
```

4.

```
    sudo useradd --shell /usr/bin/tcsh sheaff
```

5.

```
enscript -T 4 -b'ECE 331 Homework 5 %E %*|$%|John Bowen' -O hw05 -o - | ps2pdf - hw05.pdf
```