Nicholas LaJoie, ECE 331, HW 5

```
// Author: Nicholas LaJoie
// ECE 331 - Homework 5
// Date: February 23, 2017

1. C-Style Floating Point Match
    a. Minimum Match: 0. or .0
    b. Regex: ((\d*\.\d+)|(\d+\.)([eE][-+]?\d+)?[flFL]?)|(\d+[eE][-+]?\d+[flFL]?)/;

2. Morse Code Characters

#!/usr/bin/perl
# Author: Nicholas LaJoie
# ECE 331 - Homework 5, Problem 2 - Version2
# Date: February 20, 2017
# File: morse_encode.pl
# Description: Perl script converts ASCII Morse data to C source and header files
# Input: morse_chars.txt
# Output: encoding.c encoding.h
# Sources: perldoc.perl.org, perlmaven.com/how-to-sort-a-hash-in-perl

use POSIX;

# Open input file
open(IN, "morse_chars.txt") or die "Cannot open \"morse_chars.txt\", must be in the current di
rectory.\n";

# Open output files
open(my $outc, ">", "encoding.c") or die "Failed to create \"encoding.c\"\n";
open(my $outh, ">", "encoding.h") or die "Failed to create \"encoding.h\"\n";

# Predefined strings
my $header_comments = "// Author: Nicholas LaJoie\n// ECE 331 - Homework 5\n// Date: February
20, 2017\n";

# Create encoding.h file
print $outh "$header_comments// File: encoding.h\n// Description: Header file for encoding mor
se code in C\n\n";
print $outh "#ifndef ENCODING_H\n#define ENCODING_H\n\n";
print $outh "struct morse_s {\n\tunsigned int bin;\n\tint len;\n};\n\n";
print $outh "extern struct morse_s list[256];\n\n";
print $outh "#endif\n";

# Create encoding.c file
print $outc "$header_comments// File: encoding.c\n// Description: Implementation for encoding
morse code in C\n\n";
print $outc "#include <stdio.h>\n#include \"encoding.h\"\n\n";
print $outc "struct morse_s list[256] = {\n";

# Create a hash of characters => binary encoding (in reverse)
# Matches valid input lines (<Character> <Morse Code Encoding>)
my %encoding;
my $bin;
while (<IN>) {
    chomp;
    if (/(^.{1})\s([\s*-]+$)/) {
        my $rev = reverse $2;
        foreach $c (split //, $rev) {
            if ($c =~ m/[*-]/) {
                $bin .= "1";
            } else {
                $bin .= "0";
            }
        }
        $encoding{$1} = "0b$bin";
        $bin = "";
    }
}

# Sort hash by value of char
foreach my $char (sort (keys(%encoding))) {}

# Print to encoding.c
# If the hash exists, encode it, else encode a 0b0 with length 0 (covers all 256 bytes)
for (my $i = 0; $i < 256; $i++) {
    if (exists $encoding{chr($i)}) {
        print $outc "\t{$encoding{chr($i)}, " . (length($encoding{chr($i)}) - 2) . "},\n";
    } else {
        print $outc "\t{0b0, 0}";
        if ($i != 255) {
            print $outc ",\n";
        } else {
            print $outc "\n";
        }
    }
}

# Conclude struct array initialization
print $outc "};\n";

# Close files
close IN;
close $outh;
close $outc;
```

3. Print Morse

```
// Author: Nicholas LaJoie
// ECE 331 - Homework 5, Problem 3
// Date: February 19, 2017
// Description: Receives a valid string as a command line argument, encodes it in morse code,
and prints to the console
// Sources: http://stackoverflow.com/questions/18921559/initializing-array-of-structures, http
://stackoverflow.com/questions/8206014/passing-an-array-of-structs-in-c

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "encoding.h"

void print_morse(const struct morse_s x);

int main (int argc, char * argv[])
{
    int i;

    // Verify command line argument
    if (argc != 2) {
        printf("Usage: %s \"<Input String>\"\n", argv[0]);
        return 1;
    }

    // Process input and print morse encoding
    char *str = argv[1];
    for (i = 0; i < strlen(str); i++) {
        // Each encoding can be accessed using list[str[i]]
        print_morse(list[(int)toupper(str[i])]);
        // Print three Morse time units between characters (not at the end, however)
        if ((i != (strlen(str) - 1)) && (str[i + 1] != ' ') && (str[i] != ' ')) {
            printf("___");
        }
    }
    // Finish with a newline
    printf("\n");
    return 0;
}

// Function prints morse_s struct in morse code ('*' for 1, '_' for 0)
void print_morse(const struct morse_s x)
{
    int i, mask = 0b1;

    // Print the binary value using a mask
    for (i = 0; i < x.len; i++) {
        printf("%c", ((x.bin & mask) == 0) ? '_':'*');
        mask <<= 1;
    }
}
```

4. Makefile

```
TARGET=morse_encode
CFLAGS=-g -Wall
OBJS=morse_encode.o encoding.o
LIBS=
CC=gcc

.PHONY: all clean

all: ${TARGET}

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

clean:
        rm -f ${TARGET} ${OBJS}
```

5. Preamble

```
// Author: Nicholas LaJoie
// ECE 331 - Homework 5, Problem 5
// Date: February 19, 2017
// Description: Receives a valid string as a command line argument, encodes it in morse code,
and prints to the console - includes a preamble
// Sources: http://stackoverflow.com/questions/18921559/initializing-array-of-structures, http
://stackoverflow.com/questions/8206014/passing-an-array-of-structs-in-c

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "encoding.h"

void print_morse(const struct morse_s x);

int main (int argc, char * argv[])
{
    int i;
    struct morse_s preamble = {0b0100, 4};

    // Verify command line argument
    if (argc != 2) {
        printf("Usage: %s \"<Input String>\"\n", argv[0]);
        return 1;
    }

    // Print preamble
    print_morse(preamble);
    // Process input and print morse encoding
    char *str = argv[1];
    for (i = 0; i < strlen(str); i++) {
        // Each encoding can be accessed using list[(int)str[i]]
        print_morse(list[(int)toupper(str[i])]);
        // Print three Morse time units between characters (not at the end, however)
        if ((i != (strlen(str) - 1)) && (str[i + 1] != ' ') && (str[i] != ' ')) {
            printf("___");
        }
    }
    // Finish with a newline
    printf("\n");
    return 0;
}

// Function prints morse_s struct in morse code ('*' for 1, '_' for 0)
void print_morse(const struct morse_s x)
{
    int i, mask = 0b1;

    // Print the binary value using a mask
    for (i = 0; i < x.len; i++) {
        printf("%c", ((x.bin & mask) == 0) ? '_':'*');
        mask <<= 1;
    }
}
```

6. Checksum

```
// Author: Nicholas LaJoie
// ECE 331 - Homework 5, Problem 6
// Date: February 19, 2017
// Description: Receives a valid string as a command line argument, encodes it in morse code,
and prints to the console - includes a preamble and 8-bit checksum
// Sources: http://stackoverflow.com/questions/18921559/initializing-array-of-structures, http
://stackoverflow.com/questions/8206014/passing-an-array-of-structs-in-c

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "encoding.h"

int print_morse(const struct morse_s x);

int main (int argc, char * argv[])
{
    int i, sum = 0;
    struct morse_s preamble = {0b0100, 4};
    struct morse_s checksum;

    // Verify command line argument
    if (argc != 2) {
        printf("Usage: %s \"<Input String>\"\n", argv[0]);
        return 1;
    }
    // Print preamble
    sum += print_morse(preamble);
    // Process input and print morse encoding
    char *str = argv[1];
    for (i = 0; i < strlen(str); i++) {
        // Each encoding can be accessed using list[(int)str[i]]
        sum += print_morse(list[(int)toupper(str[i])]);
        // Print three Morse time units between characters (not at the end, however)
        if ((i != (strlen(str) - 1)) && (str[i + 1] != ’ ’) && (str[i] != ’ ’)) {
            printf("___");
        }
    }
    // Finish it off with the 0, checksum, and newline
    printf("_");
    checksum.bin = ˜sum;
    checksum.len = 8;
    print_morse(checksum);
    printf("\n");

    return 0;
}

// Function prints morse_s struct in morse code (’*’ for 1, ’_’ for 0)
// Returns the number of ’1’s printed (used for the checksum)
int print_morse(const struct morse_s x)
{
    int i, ones = 0, mask = 0b1;

    // Print the binary value using a mask
    for (i = 0; i < x.len; i++) {
        if ((x.bin & mask) != 0) {
            printf("*");
            ones++;
        } else {
            printf("_");
        }
        mask <<= 1;
    }
    return ones;
}
```

7. Sysfs

```
// Author: Nicholas LaJoie
// ECE 331 - Homework 5, Problem 7
// Date: February 21, 2017
// Description: Receives a valid string as a command line argument, encodes it in morse code,
and prints to the console - includes a preamble and 8-bit checksum
// BPSK Encoding: Uses GPIO 4 as Enable Output, GPIO 17 as Morse Code data BPSK encoded
// Note: Utilizing code I had written for 471 last semester
// Sources: http://stackoverflow.com/questions/18921559/initializing-array-of-structures, http
://stackoverflow.com/questions/8206014/passing-an-array-of-structs-in-c, http://elinux.org/RPi
_GPIO_Code_Samples

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "encoding.h"
#include <fcntl.h>  /* open() */
#include <unistd.h> /* close() */

#define MORSE_TIME_UNIT 120000 // microseconds
#define ENABLE_PIN 4
```

Nicholas LaJoie, ECE 331, HW 5

```
#define BPSK_PIN 17

void pin_high(int fd, const int pin);
void pin_low(int fd, const int pin);
int bpsk_encode(int fd, const struct morse_s x);

int main (int argc, char * argv[])
{
    int i, fd, sum = 0;
    char buffer[4006];
    struct morse_s preamble = {0b0100, 4};
    struct morse_s three_units = {0b000, 3};
    struct morse_s one_unit = {0b0, 1};
    struct morse_s checksum;

    // Verify command line argument
    if (argc != 2) {
        printf("Usage: %s \"<Input String>\"\n", argv[0]);
        return 1;
    }

    /* GPIO Set Up --------------------------------------------------------------*/
    // Enable GPIO Pin 4
    fd = open("/sys/class/gpio/export", O_WRONLY);
    if (fd < 0) {
        perror("Error enabling GPIO 4\n");
        return 1;
    }
    sprintf(buffer, "4");
    write(fd, buffer, strlen(buffer));
    close(fd);

    // Set GPIO Pin 4 direction to out
    fd = open("/sys/class/gpio/gpio4/direction", O_WRONLY);
    if (fd < 0) {
        perror("Error setting direction of GPIO 4\n");
        return 2;
    }
    sprintf(buffer, "out");
    write(fd, buffer, strlen(buffer));
    close(fd);

    // Enable GPIO Pin 17 and set to out
    fd = open("/sys/class/gpio/export", O_WRONLY);
    if (fd < 0) {
        perror("Error enabling GPIO 17\n");
        return 3;
    }
    sprintf(buffer, "17");
    write(fd, buffer, strlen(buffer));
    close(fd);

    // Set GPIO Pin 17 direction to out
    fd = open("/sys/class/gpio/gpio17/direction", O_WRONLY);
    if (fd < 0) {
        perror("Error setting direction of GPIO 17\n");
        return 4;
    }
    sprintf(buffer, "out");
    write(fd, buffer, strlen(buffer));
    close(fd);

    // Enable line (GPIO 4) is active low, set high
    pin_high(fd, ENABLE_PIN);

    // Set Morse encode line (GPIO 17) to low
    pin_low(fd, BPSK_PIN);
    /* Completed GPIO Set Up ----------------------------------------------------*/

    /* Encoding Process */
    // Set Enable Low (1 morse time unit)
    pin_low(fd, ENABLE_PIN);
    printf("ENABLE SET LOW\n");
    usleep(MORSE_TIME_UNIT);

    // Encode preamble
    sum += bpsk_encode(fd, preamble);

    // Encode input
    char *str = argv[1];
    for (i = 0; i < strlen(str); i++) {
        // Each encoding can be accessed using list[(int)str[i]]
        sum += bpsk_encode(fd, list[(int)toupper(str[i])]);
        // Encode three Morse time units between characters (not at the end, however)
        if ((i != (strlen(str) - 1)) && (str[i + 1] != ' ') && (str[i] != ' ')) {
            bpsk_encode(fd, three_units);
        }
    }

    // Finish it off with the 0, checksum, and newline
    bpsk_encode(fd, one_unit);
    checksum.bin = ~sum;
    checksum.len = 8;
    bpsk_encode(fd, checksum);
    printf("\n");

    // Return BPSK_Pin to low
    pin_low(fd, BPSK_PIN);

    // Set Enable High (Encoding complete)
    usleep(MORSE_TIME_UNIT);
    pin_high(fd, ENABLE_PIN);
    printf("ENABLE SET HIGH\n");
    /* Encoding complete */

    return 0;
}

// Function sets the given GPIO pin high
void pin_high(int fd, const int pin)
{
   char buf[10];
   char path[30];

   snprintf(path, 30, "/sys/class/gpio/gpio%d/value", pin);
   fd = open(path, O_WRONLY);
   if (fd < 0) {
       printf("Error setting pin %d high.\n", pin);
       return;
   }
   sprintf(buf, "1");
   write(fd, buf, strlen(buf));
   close(fd);
}

// Function sets the given GPIO pin low
```

Nicholas LaJoie, ECE 331, HW 5

```
void pin_low(int fd, const int pin)
{
   char buf[10];
   char path[30];

   snprintf(path, 30, "/sys/class/gpio/gpio%d/value", pin);
   fd = open(path, O_WRONLY);
   if (fd < 0) {
       printf("Error setting pin %d low.\n", pin);
       return;
   }
   sprintf(buf, "0");
   write(fd, buf, strlen(buf));
   close(fd);
}

// Function encodes a character and returns the number of ’1’s encoded (used for the checksum)
int bpsk_encode(int fd, const struct morse_s x)
{
    // Track phase and last value (0 or 1)
    // Phase 0 -> low to high, Phase 1 -> high to low
    int i, cur_val, mask = 0b1, ones = 0;
    static int phase = 0;

    for (i = 0; i < x.len; i++) {
        // Get current value
        if ((x.bin & mask) == 0) {
            cur_val = 0;
        } else {
            cur_val = 1;
            ones++;
        }

        // If current value is 1, toggle phase
        if (cur_val == 1) {
            phase ^= 1;
        }

        // Print based on phase
        if (phase == 0) {
            // Go low
            pin_low(fd, BPSK_PIN);
            usleep(MORSE_TIME_UNIT / 2);
            // Go high
            pin_high(fd, BPSK_PIN);
            usleep(MORSE_TIME_UNIT / 2);
        } else if (phase == 1) {
            // Go high
            pin_high(fd, BPSK_PIN);
            usleep(MORSE_TIME_UNIT / 2);
            // Go low
            pin_low(fd, BPSK_PIN);
            usleep(MORSE_TIME_UNIT / 2);
        }
        // Shift mask
        mask <<= 1;
    }

    // Completed character encoding
    return ones;
}

--------- Updated Makefile ---------
```

Nicholas LaJoie, ECE 331, HW 5

```
TARGET=bpsk_encode
CFLAGS=-g -Wall
OBJS=bpsk_encode.o encoding.o
LIBS=
CC=gcc

.PHONY: all clean

all: ${TARGET}

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

clean:
	rm -f ${TARGET} ${OBJS}
```