COMP 211: Computer Science I
Homework 4: Functions in C

PROBLEM 1. *A* checksum *is a value computed from some data that is used to verify its validity. For example, a bank might only use account numbers $d_0d_1d_2\ldots$ such that*

$$d_0 + f(d_1) + d_2 + f(d_3) + \cdots \bmod 10 = 0,$$

where $f(n)$ is the sum of the digits in the 2-digit base-10 representation of $2n$. The value $d_0 + f(d_1) + d_2 + f(d_3) + \cdots$ is called a checksum*. For example, 17327 would be a valid account number because $1+f(7)+3+f(2)+7 = 1+5+3+4+7 = 20$ and $20 \bmod 10 = 0$ ($a \bmod b$ is the remainder upon dividing a by b, so we are really saying that the sum is divisible by 10).*

The reason for using a checksum like this is that it allows the bank to check for errors when account numbers are transmitted electronically. If there is a transmission error that changes some of the digits in the account number, it is unlikely that the resulting digits satisfy this property. For example, if you change one digit of 17327, you will get a number whose checksum is not divisible by 10, and it is unlikely that if you transpose two digits of a valid account number that you will end up with a valid account number.

Your first task is to write a program that asks the user to enter a possible account number, which you store as an `unsigned` `long`*, and then reports whether or not it is valid by using the technique described above. You must define and use at least the following functions:*

- `max_pow10`$(n) = k$, *where* $10^k \le n < 10^{k-1}$.
- `sum_digits`(n) = *the sum of the 2-digit decimal representation of $2n$. For example,* `sum_digits`$(7) = 5$ *because* $2 \cdot 7 = 14$ *and* $1 + 4 = 5$.
- `checksum`$(acctnum)$ = *the checksum for acctnum as described above.*
- `is_valid`$(acctnum)$ = `true` *if acctnum is a valid account number,* `false` *otherwise.*

Your second task is to write a program that asks the user to enter a number n with an even number of digits not starting with 0, and then figure out a digit d such that the number obtained by following n by d is a valid account number. For example, if the user entered 1732 for n, your program would compute 7 for d. You should write another appropriate function for computing the final digit that uses `checksum`*.*

You third task (and what you will actually submit) is to write a program that first asks the user whether they want to validate an account number or create one. If the former, ask the user to enter an account number and report whether or not it is valid (the account number may have any number of digits, but you can assume the first digit is not 0) as in your first task. If the latter, ask the user to enter a number with an even number of digits, not starting with 0, then compute a final digit d as in your second task and report the valid account number that consists of the number the user entered followed by d. Figure 1 shows a sample trace from some runs of my solution.

The format code for reading a character from the user with `scanf` *is* `"%c"`*. However, there is a slight complication: the format string argument must begin with a space character. Thus, to read a character from the terminal and store it in the variable* `response`*, you would call*

```
scanf(" %c", &response) ;
```

Name your program `hw4_accnum.c`*.*

This problem is adapted from Exercise 2.1.15 of Sedgewick, R. and Wayne, K., Introduction to Programming in Java: An Interdisciplinary Approach*, Addison-Wesley, 2006.*

```
$ cc -Wall -o hw4_acctnum hw4_acctnum.c
$ ./hw4_acctnum
(V)alidate or (C)onstruct? V
Enter an account number: 17327
      17327 is a valid account number.
$ ./hw4_acctnum
(V)alidate or (C)onstruct? V
Enter an account number: 18327
      18327 is not a valid account number.
$ ./hw4_acctnum
(V)alidate or (C)onstruct? V
Enter an account number: 1732767
    1732767 is a valid account number.
$ ./hw4_acctnum
(V)alidate or (C)onstruct? C
Enter a prefix with an even number of digits: 1732
    Constructed account number: 17327.
$ ./hw4_acctnum
(V)alidate or (C)onstruct? C
Enter a prefix with an even number of digits: 1832
    Constructed account number: 18325.
$ ./hw4_acctnum
(V)alidate or (C)onstruct? C
Enter a prefix with an even number of digits: 173276
    Constructed account number: 1732767.
```

Figure 1: Sample trace from `hw4_acctnum.c`*.*

PROBLEM 2. *The barcode used by the U.S. Postal System to route mail is defined as follows. Each decimal digit in the zip code is encoded using a sequence of three half-height and two full-height bars, as shown in Figure 2. The barcode starts with a full-height bar followed by the bars for each digit in the zip code, followed by the bars for a checksum digit, followed by a full-height bar. The first and last full-height bars are called the* guard bars. *The checksum digit for the zip code* $d_0d_1d_2d_3d_4$ *is defined to be* $d_0 + d_1 + d_2 + d_3 + d_4 \bmod 10$. *Your task is to ask the user to enter a five-digit zip code and then print the corresponding bar code to the terminal. You must define and appropriately use the following functions:*

- `draw_bar`(hh): *print either a half-height (if* $hh =$ `true`*) or full-height (if* $hh =$ `false`*) bar to the terminal. Draw the bar horizontally on one line.*
- `draw_bar_digit`(n): *print the encoding for the number* n. *You may assume that* $0 \leq n < 10$. *Print the bars on successive lines of the terminal.*
- `checksum`(n) = *the checksum for* n, *where* n *is assumed to be a 5-digit zip code.*

Write and use other functions as appropriate. To show that you understand function prototypes/declarations, define all of these functions after *you define* `main` *(so you will have to write prototypes/declarations before you define* `main`*). Ensure that there is a blank line after the first guard and before the end guard, and also between each group of five bars (this just makes it easier for grading).*

```
0:                                      5:
********                                ****
********                                ********
****                                    ****
****                                    ********
****                                    ****
1:                                      6:
****                                    ****
****                                    ********
****                                    ********
********                                ****
********                                ****
2:                                      7:
****                                    ********
****                                    ****
********                                ****
****                                    ****
********                                ********
3:                                      8:
****                                    ********
****                                    ****
********                                ****
********                                ********
****                                    ****
4:                                      9:
****                                    ********
********                                ****
****                                    ********
****                                    ****
********                                ****
```

Figure 2: Encodings of the digits 0–9.

You should declare global constants for the symbol (a <u>`char`</u> *value) out of which to draw the bars, and the number of symbols to draw for a half-height bar (or for a full-height bar). Zip codes should be stored as* <u>`unsigned`</u> <u>`int`</u> *values. A sample trace of some runs from my solution are in Figure 3, where I use* `'*'` *for the symbol out of which to build the bars and define a half-height bar to consist of* 3 *characters.*

Name your program `hw4_postal.c`*.*

This problem is adapted from Exercise 2.1.37 of Sedgewick, R. and Wayne, K., Introduction to Programming in Java: An Interdisciplinary Approach*, Addison-Wesley, 2006.*

Submission

Submit the files `hw4_acctnum.c` and `hw4_postal.c`.

```
$ ./hw4_postal
Enter 5-digit ZIP code: 06457
The postal bar code for 06457 is:
********

********
********
****
****
****

****
********
********
****
****

****
********
****
****
********

****
********
****
********
****

********
****
****
****
********

****
****
********
****
********

********
```

```
$ ./hw4_postal
Enter 5-digit ZIP code: 94530
The postal bar code for 94530 is:
********

********
****
********
****
****

****
********
****
****
********

****
********
****
********
****

****
****
********
********
****

********
********
****
****
****

****
****
****
********
********

********
```

Figure 3: Sample traces from `hw4_postal.c`.