EECS 2032

LAB 7
Winter 2020

In this lab, you are introduced to file and I/O operations in C and some simple data manipulation techniques.

Problem 1 Calculating a Checksum

When data are transmitted it might get corrupted during the transmission. One way to check if the data is corrupted or not, is to introduce ***checksum*** which is derived from the data, if data are changed (corrupted), it produces a different checksum and it will be easy to detect the corruption (not correct it though).
One way of doing so is the ***1's complement addition*** used in TCP/IP.
Data is divided into chunks of 16-bit each. All the 16-bit chunks are added in 1's complement addition. After adding all chunks, we take the ***1's complement*** of the final sum. That produces a 16-bit checksum. The checksum is appended to the end of the data.
To check if the data is corrupted or not, we divide the data into 16-bit chunks (including the checksum we added to the end of the transmitted data), we add all the chunks in a 1's complement fashion. If the final result is a number made of all 1's, then there was no corruption, else the data have changed.
Now, we explain some of the terms we used in the above discussion.

Checksum

A fixed length data added to the end of the data transmitted in order to be able to detect if the data changed or not.

1's complement

Basically this means we invert every bit in the data. Each 1 becomes 0, and each 0 becomes 1. For example (the underscore to make it easy to read)

the 1's complement of `1011_0100_1010_1011`

is `0100_1011_0101_0100`

1's complement addition

In 1's complement addition, we add the two 16-bit chunks in the usual manner. If there is no carry, then the result is the 1's complement addition.

If there is a carry, then we add 1 to the number, and the carry is ignored (we add the carry to the number)

Example

```
     0101_1001_1011_1100
+
     0010_0101_1000_1000
-------------------------
     0111_1111_0100_0100
```

Since there is no carry, then this is the 1's complement addition

Example

```
     1110_0000_0101_1010
+
     0100_0101_1000_0100
-------------------------
    10010_0101_1101_1110
```

There is a carry, we remove the carry and add it to the least significant bit

```
    10010_0101_1101_1110
                       1
-------------------------
     0010_0101_1101_1111
```

This is the 1's complement addition of the above two numbers.

Specifications

- Data are read from a file called dat.txt
- Assume the file contains an even number of bytes
- Data and checksum are written to result.txt

submit as l7a.c to LAB7

Disclaimer

There are many subtleties we did not discuss in the above lab, for example the endianess of your system will affect how to read data format file and treat it as a binary number. Since we are using the same platform for both calculating the checksum and verifying it, we need not to worry about endianess.

Problem 2 Checking a checksum

In this part, you will check if the received data is corrupted or not (check if the checksum is valid or not).

After receiving the data, the data is divided into 16-bit chunks, we add all the chunks together in a 1's complement fashion (including the last chunk which is the checksum).
If the result of the addition is a number made of all 1's (0xFFFF or 65535), there is no corruption, otherwise, data is changed.

Specifications

- Data are read from a file called result.txt (the same file produced by part 1)
- The output is one word followed by a new line character
- The output is either “Valid” or “Invalid”

submit as check.c to LAB7