CS380 — Exercise 3

January 18, 2017

Due: Monday, January 23, 2017 before midnight (50 points)

You are allowed to work with a partner to complete this exercise. If you do work with a partner, only one person should create the codebank project and add the other as a member with at least developer privilege. You should also include a comment at the top of any source code files with both partners' names to ensure grades are entered correctly.

Preparing the Project

1. Go to `https://codebank.xyz` and create a new project named `CS380-EX3`.
2. On your local machine, from a terminal or git bash navigate to the folder you use for storing CS380 related files and create a new directory to store this exercise. I'd recommend calling it `CS380-EX3` to match the repository. From now on, we'll call this directory the *working directory*.
3. `cd` in to the working directory and run:

   ```
   $ git init
   $ git remote add origin https://codebank.xyz/username/CS380-EX3.git
   ```

   where `username` is your bronconame.
4. Now, the directory on your machine is a git repository with a reference to the remote repository on `https://codebank.xyz`.
5. As an alternative to creating a local repository yourself, you can `git clone` the empty repository from the website to accomplish the same thing.

The Program

Description

1. Create a Java source file[1] named `Ex3Client.java` with a class named `Ex3Client` that contains the `main` method. You can also create any other classes or files as needed.
2. Your program should create a `Socket` connection to `codebank.xyz` port number 38103.
3. In this program, I will first send you a single byte of data. Treat it as an unsigned number in the range [0, 255]. The value of this byte corresponds to the number of bytes I will be sending next.

   For example, if the first byte you read has value 40, I am going to be sending you 40 more bytes. If it is 102, I will be sending 102 more bytes.

[1] If you want to use a different language, let me know.

4. Based on the value of the previous byte I sent, read in the rest of the bytes I send and store them in an array.
5. Write a method with the following header:

   ```
   public static short checksum(byte[] b)
   ```

   This method will implement the Internet checksum algorithm that is also used in project 3 for IPv4 packets.

   The algorithm maintains a 32 bit number as the sum and goes through the byte array two bytes at a time, forms a 16-bit number out of each pair of bytes and adds to the sum. After each time it adds, it checks for overflow. If overflow occurs, it is cleared and added back in to the sum (acting like a wrap-around). Finally, when the sum is calculated we perform ones' complement and return the rightmost 16 bits of the sum.

   The C code for the algorithm is provided below:

   ```
   u_short cksum(u_short *buf, int count)
   {
       register u_long sum = 0;

       while (count--)
       {
           sum += *buf++;
           if (sum & 0xFFFF0000)
           {
               /* carry occurred.  so wrap around */
               sum &= 0xFFFF;
               sum++;
           }
       }
       return ~(sum & 0xFFFF);
   }
   ```

6. You will pass the bytes read in step 4 to the checksum method and calculate the correct checksum.
7. Finally, send this checksum as a sequence of two bytes to the server as you did for the CRC value in exercise 2.
8. The server will send a final response of either 0 or 1: if it is 1, your program worked correctly and if it's 0 then you did not calculate the correct checksum.

Sample Output

Here are some sample outputs from running the program:

```
$ java Ex3Client
Connected to server.
Reading 133 bytes.
Data received:
   EB06B6EAC96C164C2C00
   FCE2CBEF4F3394C357AD
   A76BFA05CA9985441EBB
   36984707623788A21A6A
   25C02BC59C025DAF24D4
```

```
   0281CC976EB372FA9529
   170BF60DEF43D87F8527
   97D2030C0A029F5F5AB2
   48133826A28055D29314
   B4ED480D391C1F2C88FC
   4F5092E9E3E1DF703057
   BA8E65E370E39CBF2148
   85467D45BF264B2207DE
   08E55C
Checksum calculated: 0x8342.
Response good.

$ java Ex3Client
Connected to server.
Reading 210 bytes.
Data received:
   34492FFDD26689239C9A
   5C3F8EDDBFADC0DA166C
   5730702B13A1C2836507
   46D9EBFDE4DCC95283FB
   FCF05A966468DB03E1CB
   354116EB49F2A511D548
   F89DEB1763527ABAA337
   B9257CBA1EBAC7BCD457
   EF9E9E3FA3AE1996D5CC
   6D4125C13CA01DD862E1
   ACA5F674EAA549C7CC12
   69C44B09E23D875EA863
   230D4919CC0FDE787C84
   476E478DB1959842CA84
   C0D2776A7DFE5DEE9CAB
   2625D3F13E4542A88C35
   49C9B1AA68ADE663DA0C
   FEDBA0E7BCB4AED0B129
   92EAD1206BD32946CF22
   1B492F3DD12C1B52A55D
   898A58C99FE192382CE1
Checksum calculated: 0x49C7.
Response good.

$ java Ex3Client
Connected to server.
Reading 39 bytes.
Data received:
   5ADC02B4D34929690976
   B733924D86191EAF4FB8
   EF4482274277E3D34FF5
   9CA8795D87A642284D
Checksum calculated: 0x49C2.
Response good.
```

```
$ java Ex3Client
Connected to server.
Reading 6 bytes.
Data received:
   F08F7807236C
Checksum calculated: 0x73FC.
Response good.

$ java Ex3Client
Connected to server.
Reading 146 bytes.
Data received:
   F01AD8FB947DC00684EB
   27535C1D91637B16F41C
   DC8F14F4F11997F815F5
   2DEFF482109E6C5E949F
   07E5EF95B366CC2E9288
   6B8E0B9B93CA89626A19
   68AD8380CF2413AF7EFD
   D68E1ED6C9285C539A8E
   7E31AE48093FF8DF8C60
   9762D282DB6A2349E67A
   35CA0FD426EEFBC289C5
   8DB67E1C9B18D222E3C0
   8318A61D64D60B3C690D
   7DD53F4A432170EFBFFB
   4C69E61BE322
Checksum calculated: 0x6743.
Response good.
```

Submission

You can use `git add`, `git commit`, and `git push` to push the changes to https://codebank.xyz. You can make as many commits and push as many times as desired before the deadline.