Question 1: Hash Function Collisions (24 points)

We saw that collisions can be more powerful depending on the file format. Some formats by virtue of their design can have more devastating effects for an arbitrary collision. Imagine the following bizarrely designed file format that renders it quite vulnerable to collision attacks.[1]

```
byte  |  meaning
------+-----------------------------------
 00   | magic number fixed 0xd1
 01   | magic number fixed 0x31
 02   | magic number fixed 0xdd
 03   | short "length" high value
 04   | short "length" low value
05-18 | reserved
 19   | short "jump" high value
 20   | short "jump" low value
 ..   |
 J+0  | short "datalength" high value
 J+1  | short "datalength" low value
 J+2  | byte 0 for stored data
 J+3  | byte 1 for stored data
 ..   |  ..
J+1+dl| byte dl-1 for stored data (last)
------+-----------------------------------

J = jump MOD length
dl = datalength
```

The first three bytes are a "magic number", which typically is used to indicate the file format; in this case the byte sequence `0xd131dd` is used. (The tool `file` can tell you what type of file some blob is in part by using this magic numbers, though it will of course not recognize this ad hoc format.) The next 2 bytes are a big-endian short (16-bit) value that indicates the length of the file. The next bunch of bytes are reserved for future use (another typical feature of file formats or network protocols). Then bytes 19 and 20 are another big-endian short that indicates the "jump" or offset in the file where the data is actually stored. If this jump value is larger then the file size (bytes 3 and 4) then the jump is reduced modulo the file size.

The data is stored as a length-prefixed string starting at the offset jump (modulo file size). The first two bytes of jump indicate the length of the data, and the bytes after that store the actual data.

A C++ program that "prints" the stored data for this file format is available on the course website:

[1] One may ask "why would a file format exist with such a bizarre semantics and which renders it so particularly vulnerable to hash collisions" and indeed it is rather contrived, nevertheless it should not be the concern for file format designers to consider susceptibility for hash collisions in the designs but rather the concern of hash function designers to avoid having collisions.

`https://pages.cpsc.ucalgary.ca/~joel.reardon/526/assn/fileformat.cc`

You can compile this and run it, passing a filename as the first argument. If the file is not correctly formatted, then it will hopefully print a useful error message or otherwise just fail. Feel free to modify it to help you create a well-formatted file, but you should double check your result with the original program to ensure it is working the same. Some example files that show the format are available here:

`https://pages.cpsc.ucalgary.ca/~joel.reardon/526/assn/ex1.sff`
`https://pages.cpsc.ucalgary.ca/~joel.reardon/526/assn/ex2.sff`
`https://pages.cpsc.ucalgary.ca/~joel.reardon/526/assn/ex3.sff`
`https://pages.cpsc.ucalgary.ca/~joel.reardon/526/assn/ex4.sff`

Your submission for this question is submit two different files. These two files must have the same md5 hash, and you will find the md5 collision useful (indeed, what an unfortunate choice the file format designed had for the magic number). The collisions are available here:

`https://pages.cpsc.ucalgary.ca/~joel.reardon/526/assn/coll1`
`https://pages.cpsc.ucalgary.ca/~joel.reardon/526/assn/coll2`

These files are storing the following collisions, represented in hexadecimal with added whitespace:

```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb
7f8955ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf2
80373c5bd8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd0239
6306d248cda0e99f33420f577ee8ce54b67080a80d1ec69821bcb6a88393
96f9652b6ff72a70
```

and

```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb
7f8955ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd72
80373c5bd8823e3156348f5bae6dacd436c919c6dd53e23487da03fd0239
6306d248cda0e99f33420f577ee8ce54b67080280d1ec69821bcb6a88393
96f965ab6ff72a70
```

You can download these files and confirm that they have the same MD5 hash with `md5sum` yet differ with `diff`. Using these, perform a length extension attack to have two different files that appear to "store" different data when read with a tool suited to their format.

The "stored data" for one file shall be "<your ucid> will receive an A+ in CPSC 526" and the other shall be "<your ucid> will receive a Z- in CPSC 526". You can use the C++ program to check the output is correct and `md5sum` to ensure that they have the same hash.

Question 2: PRNG (12 points)

In class we saw two properties of cryptographically-secure pseudorandom number generators: prediction resistance and rollback resistance.

Describe two approaches that use a small (e.g., 256-bit) seed to generate a stream of randomness that

1. achieves rollback resistance *but not* prediction resistance
2. achieves prediction resistance *but not* rollback resistance

That is, provide two different approaches that each achieve exactly one of the two desired properties. You may assume standard cryptographic assumptions hold. It is 6 points for approach: 2 points for the description of PRNG, 2 points for why it either has or doesn't have each of the two properties.

Be sure to express your design clearly (i.e., use pseudocode if necessary). You may use basic cryptographic functions but just define what they mean. You can use tutorial exercise style pseudocode as a good idea as to expectations.

Submit your written answer to the assignment 2 dropbox on D2L.

Question 3: Kerberos Forward Secrecy (12 points)

Kerberos does not have forward secrecy as provided for communications between Alice (the user) and Bob (e.g., the printer). But forward secrecy can still be added after Alice and Bob have mutually authenticated.

1. What are the long-term secrets in Kerberos? (2 points)
2. What are the short-term secrets in Kerberos? (2 points)
3. Explain why Kerberos does not have forward secrecy. Recall that forward secrecy is the property that the leak of a long-term secret does not reveal the contents of communication sessions that were secured by that long-term secret. Be specific about what data needs to be compromised and what the consequences of it are. (4 points)
4. Augment Kerberos to have forward secrecy for the actual communication between Alice and Bob. You only need to specify the message sequences for the parts of the Kerberos protocol that you change (i.e., if Alice's communication with the TGS is unchanged then you do not need to specify it). (4 points)