1. (40 points) In this problem, our hash function $H$ is the 32-bit truncated SHA-256. That is, $H(x)$ is the first 32 bits of SHA-256($x$) for every string $x$. You are supposed to implement the Rho method to find collision on $H$.

**Output format.** Your program needs to output two distinct strings $x$ and $x'$, their common hash output $H(x) = H(x')$, and their SHA-256 outputs SHA-256($x$) and SHA-256($x'$) in this order. If you can’t find a collision, output a failure message.

**Requirements.** Since this is a constant-memory attack, you’re not allowed to use any data structure in your code. Moreover, you have to start with a random point $x_0$, meaning that if I run your program multiple times, I should get different collisions. To avoid long waiting time, you should terminate your program with a failure message if Floyd’s tortoise-and-hare algorithm can’t detect a cycle after $2^{20}$ steps. Make sure that your program can run in linprog.

**Deliverables.** Upload to Canvas a zip file containing your source code, which includes a README.txt that informs me how I should run the program.

2. (60 points) Fix a blockcipher $E$ with an 8-byte (64-bit) block length. Given a byte string $M$, let $\text{pad}(M)$ be $M$ followed by enough bytes to take you to the next multiple eight bytes: append either the byte 01, or two bytes of 02 (that is, 02 02) or three bytes of 03 (that is, 03 03 03), and so on, up to appending eight bytes of 08 (all of these constants written in hexadecimal). Let CBC$\#$ be the variant of CBC encryption that encrypts $M$ by applying CBC, over $E$, with a random IV, to $\text{pad}(M)$. The method is specified in Internet Standard RFC 2040. Note that a CBC$\#$ ciphertext for $M$ will have the form $C = IV \parallel C'$ where $|IV| = 64$ and $|C'|$ is the least multiple of 64 exceeding $|M|$. (Recall that for a string $X$, we write $|X|$ to denote the bit length of $X$.)

a) (10 points) Write a careful fragment of pseudocode for an algorithm Decrypt to decrypt a byte string $C$ under CBC$\#$. Let Decrypt$K(C)$ return the distinguished symbol ⊥ if it is provided an invalid ciphertext; otherwise, it returns a byte string $M$.

b) (25 points) Suppose an adversary is given an oracle, Valid, that, given a ciphertext $C$, returns a single bit: the bit “1” if $C$ is valid—meaning Decrypt$K(C) \in \{0,1\}^*$—and the bit “0” if it is not—meaning Decrypt$K(C) = \bot$. Show how to use the oracle to decrypt a block $Y = E_K(X)$ for an arbitrary 8-byte $X$.

**Hint:** all your queries to the Valid oracle will be 16 bytes (namely 2 blocks). I don’t mind if you make several hundred of them.

c) (25 points) Show how to decrypt any ciphertext $C = \text{CBC}_K^8(M)$ given a Valid oracle.

**Note:** If you can’t solve part (b), you can assume that there is an adversary $A$ that does the job for part (b), and show how to use it for part (c).

3. (50 points) In this problem, if $x$ and $y$ are strings of the same length, then we write $x \sqsubseteq y$ if $x = y$ or if $x$ comes before $y$ in standard dictionary ordering.
Suppose a function $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$ has the following property. For all strings $x$ and $y$ of the same length, if $x \sqsubseteq y$ then $H(x) \sqsubseteq H(y)$. Show that $H$ is not collision resistant—describe how to efficiently find a collision in such a function.

**Hint:** The domain $\{0, 1\}^{n+1}$ will surely contain collision points, as it’s larger than the range $\{0, 1\}^n$. Treat this as a search problem over a set of $N = 2^{n+1}$ elements, and use some binary-search trick to find collision points in $O(\log(N)) = O(n)$ time.