1. (60 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. (30 points) Fix a blockcipher $E : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ and let CBCMAC$_K(M)$ be the CBC MAC, using $E_K$, of a message $M$ that is a positive multiple of $n$ bits. Let $H : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$ be the hash function such that $H(X) = \text{CBCMAC}_K^*(X)$, where $K^*$ is a public, constant key. (Note that $H$ only operates on $2n$-bit strings.) Give an attack that breaks the collision resistance of $H$ and analyze its advantage.

3. (60 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:** Binary search, always recursing on a range that is guaranteed to contain a collision.