To solve this challenge, we need to understand the principle of Game Theory named perfect play. Perfect play in Game Theory refers to a situation where players make optimal decisions at every possible point in a game, aiming to maximize their outcomes while minimizing their opponents' chances of winning. This is also the situation here, because we know that the adversary figured out the hash just from those responses, no chance involved.

For the first half of the hash, we can assume optimal strategy was used, the adversary using the minimum number of queries needed. So we know each character was only queried once, therefore for each position the correct character will only appear once since every time a character isn't queried, on its spot a different character is placed.

For the second half of the hash, we know that the characters that appear will always be different from the correct characters. Since we know the adversary found the complete hash and ignore randomness, that must mean that there is only one character which has never appeared in the other hashes on a position. So we need to find what character never appeared.

```
from collections import Counter

replies = [
    "f14fd2705fa37ce36ab73472883cf329917c50eb06d2080f863fd6bf712377b1",
    "ad3b393e6426aef4db1a49180797393a3cb6a7516d1b5f69a3d36138d9be121c",
    "73c370746eb072da5deb8ce13febaa16d33dc714c24a8424018a8a16f46cbdcf",
    "05bac2205f124251c03863a608322b5486c2ba8c1fe0fbaccb1942ed838deb30",
    "312f3b3ef4ce5ef837b1c9837ba8f9ba6f6f21f73eff19ea39e105a1604c61fe",
    "983bd1838b16b697c90731e7602c1a1ce44f4c62032a32f2dfaa50f638f14925",
    "7ec14b2e54a24a7150f121468b5dabb47dd3eb0fbb13ea33b8f7dc171d9fc8ed",
    "313f4b83fbbe56da3a34c0e13fa2e55cf5e5592344c4297b122899629e5290d3",
    "98b1c1348423aa782838204775281aff2e5432782539d3832aa2142b0ca92a34",
    "dc2bc85fa5302c51e919bb59d6eb27dc2a0004b9d867b7505e6528435a16a58a",
    "fe0e584025c1816621aa3c745971c9d9c299733e897d51d6f54cb79ac3770f08",
    "7561b7a5a2348c8bed05b7c9f6e4332718188dc8e0919d1e90bdced42be58476",
    "adc3d4612bd6215627a94b58579ba789003a6faa97566c257d56fdc9a2603290",
    "012f795543d1be84f81584c23b3123365187d52d7abe46cd44c073558534fe62",
    "734ac4a39ee5bc9b6d044c19d9e7c386b7a198d6f188ce48ef04ed0e47d8f35b",
    "3563504f92d40f67f1f8c7a4f57437274b2b1690ad0570b11c9b3f80e6cb5ca7"
]

first_half_correct_chars = {}
second_half_missing_chars = {}

for i in range(32):
    char_count = Counter(reply[i] for reply in replies)
    for char, count in char_count.items():
        if count == 1:
            first_half_correct_chars[i] = char
            break
```

```
print("Identified Correct Characters for First Half:")
for position, char in first_half_correct_chars.items():
    print(f"Position {position}: {char}")

for i in range(32, 64):
    char_count = Counter(reply[i] for reply in replies)
    for char in "0123456789abcdef":
        if char not in char_count:
            second_half_missing_chars[i] = char
            break

print("\nIdentified Missing Characters for Second Half:")
for position, char in second_half_missing_chars.items():
    print(f"Position {position}: {char}")

constructed_hash = ''.join(
    first_half_correct_chars.get(i, second_half_missing_chars.get(i, '0')) for i
in range(64)
)

print("\nConstructed Hash:", constructed_hash)
```

Made with love by: AndreiCat