```python
import csv
import sys


def main():
    print()
    # TODO: Check for command-line usage
    if len(sys.argv) != 3:
        print("Usage: python dna.py data.csv sequence.txt")

    # TODO: Read database file into a variable
    DNA_database = []
    with open(sys.argv[1], "r") as file:
        reader = csv.DictReader(file)
        for row in reader:
             DNA_database.append(row)

    # TODO: Read DNA sequence file into a variable
    file = open(sys.argv[2])
    for line in file:
        sequence = line
    file.close()

    # TODO: Find longest match of each STR in DNA sequence
    STRS = list(DNA_database[0])
    STRS.pop(0)
    repeats = {}
    for subsequence in STRS:
        longest_run = longest_match(sequence, subsequence)
        repeats.update({subsequence:longest_run})

    # TODO: Check database for matching profiles
    for person in DNA_database:
        count = 0
        for STR in STRS:
            if int(person[STR]) == repeats[STR]:
                count += 1
            else:
                break
        if count == len(STRS):
            print(person["name"])
            return

    print("No match.")
```

```
    return


def longest_match(sequence, subsequence):
    """Returns length of longest run of subsequence in sequence."""

    # Initialize variables
    longest_run = 0
    subsequence_length = len(subsequence)
    sequence_length = len(sequence)

    # Check each character in sequence for most consecutive runs of subsequence
    for i in range(sequence_length):

        # Initialize count of consecutive runs
        count = 0

        # Check for a subsequence match in a "substring" (a subset of characters) within sequence
        # If a match, move substring to next potential match in sequence
        # Continue moving substring and checking for matches until out of consecutive matches
        while True:

            # Adjust substring start and end
            start = i + count * subsequence_length
            end = start + subsequence_length

            # If there is a match in the substring
            if sequence[start:end] == subsequence:
                count += 1

            # If there is no match in the substring
            else:
                break

        # Update most consecutive matches found
        longest_run = max(longest_run, count)

    # After checking for runs at each character in seqeuence, return longest run found
    return longest_run


main()
```