```
class HashTableChaining:
    def __init__(self, size):
        self.size = size
        self.table = [[] for _ in range(size)]

    def hash_function(self, key):
        return hash(key) % self.size

    def insert(self, name, phone_number):
        index = self.hash_function(name)
        for entry in self.table[index]:
            if entry[0] == name:
                entry[1] = phone_number  # Update existing number
                return
        self.table[index].append([name, phone_number])

    def search(self, name):
        index = self.hash_function(name)
        comparisons = 0
        for entry in self.table[index]:
            comparisons += 1
            if entry[0] == name:
                return entry[1], comparisons
        return None, comparisons

class HashTableOpenAddressing:
    def __init__(self, size):
        self.size = size
        self.table = [None] * size

    def hash_function(self, key):
        return hash(key) % self.size

    def insert(self, name, phone_number):
        index = self.hash_function(name)
        while self.table[index] is not None:
            if self.table[index][0] == name:
                self.table[index][1] = phone_number  # Update existing number
                return
            index = (index + 1) % self.size
        self.table[index] = [name, phone_number]

    def search(self, name):
        index = self.hash_function(name)
```

```
        comparisons = 0
        while self.table[index] is not None:
            comparisons += 1
            if self.table[index][0] == name:
                return self.table[index][1], comparisons
            index = (index + 1) % self.size
        return None, comparisons

def compare_hash_tables(names, phone_numbers, search_names):
    # Create hash tables
    chaining_table = HashTableChaining(size=10)
    open_addressing_table = HashTableOpenAddressing(size=10)

    # Insert data into both tables
    for name, phone_number in zip(names, phone_numbers):
        chaining_table.insert(name, phone_number)
        open_addressing_table.insert(name, phone_number)

    # Search for numbers and count comparisons
    chaining_comparisons = 0
    open_addressing_comparisons = 0

    for name in search_names:
        _, comparisons = chaining_table.search(name)
        chaining_comparisons += comparisons

        _, comparisons = open_addressing_table.search(name)
        open_addressing_comparisons += comparisons

    return chaining_comparisons, open_addressing_comparisons

# Example usage
names = ["Alice", "Bob", "Charlie", "David", "Eve"]
phone_numbers = ["123-456", "234-567", "345-678", "456-789", "567-890"]
search_names = ["Alice", "Bob", "Charlie", "David", "Eve", "Frank"]

chaining_comparisons, open_addressing_comparisons = compare_hash_tables(names, phone_numbers,
search_names)

print(f"Total comparisons in Chaining: {chaining_comparisons}")
print(f"Total comparisons in Open Addressing: {open_addressing_comparisons}")
```