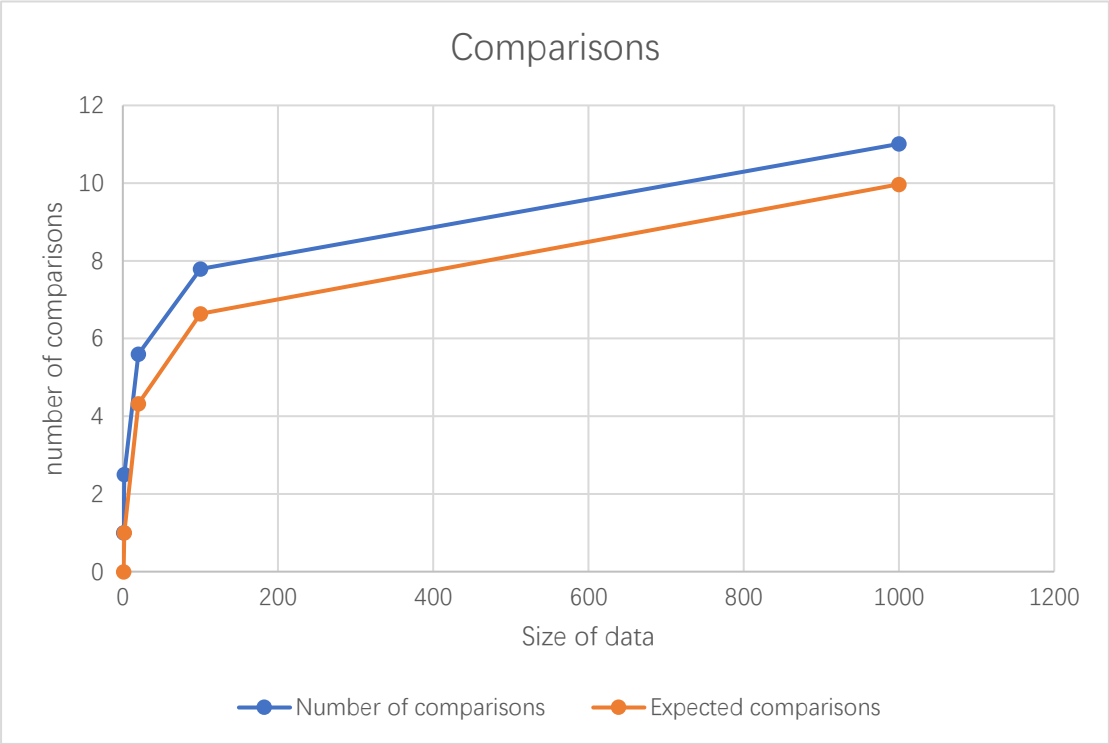
Name: Ruipu Cui
Student ID: 1298198

Introduction:
The project involves two data structures, sorted array and radix tree. The data input are stored into a struct called dataset, the trading name is the key used in comparisons and searching.

```
struct dataset{
    int census_year;
    int block_id;
    int property_id;
    int base_property_id;
    char* building_address;
    char* clue_small_area;
    char* business_address;
    char* trading_name;
    int industry_code;
    char* industry_description;
    char* seating_type;
    int number_of_seats;
    double longitude;
    double latitude;
};
```

Stage 2:
The data structure used in stage is sorted array and use binary searching and then linear searching to find the key.
The expected number of string comparisons for binary search is log(n). Here are the results obtained and result expected.

Data size	Number of comparisons	Expected comparisons
1	1	0
2	2.5	1
20	5.6	4.32
100	7.79	6.64
1000	11.01	9.97



The number of comparisons obtained is generally fit the expected comparisons.
The number of comparisons is slightly higher than the expected comparisons due to the linear search after the binary search.
The time complexity of the worst case and the average case is both O(log(n) * m) as m is the size of the query and the best case is O (1).

Stage 3.
Stage 3 use radix tree to store data. The search function that I implemented has a time complexity of O(m) and m is the number of bits of query. The algorithm of the searching function is that going the branchA when the next bit is o and going branchB when the next bit is 1 until we find a complete match.
The best, average and worst case have the same time complexity of O(m). This means that each query has the same number of bits compared in every size of data. For example, the query “Domino's Pizza” has exactly 112 bits compared in every data test. The result is match what we expected and proved the Theory..

Discussion:
The results meet the expectation. Radix tree is no doubt the better data structure when achieving Autocomplete.
First, the radix tree structure is more space efficient.as it avoids repeating store common bits.
Second, the radix tree structure is more time efficient that sorted array as the number of bits compared is significantly smaller.
This assignment enhance my understanding of the time complexity Theory as the results gives a intuitive way to compare the efficiency of different data structures.