FIT2004 S1/2019: Assignment 3

Task 1: Querying a database

The solution used in Task 1 was as follows, an array was initialized where 0-9, a-z and finally, A-Z were represented as an array length 62, with the aforementioned values directly representing spaces in the array. These spaces were then either filled with arrays if a character was found, or left empty, until the characters in both id and name were used, after which, the index was inserted at the first point of the final array in that particular id and name branch. The actual positions to fill the array were found by finding the ordinance value of the character and then adjusting it to match the array at a particular index in the array (0-9, a-z and finally, A-Z). This was recursively done for all words, this results in a Trie where each node is represented by an array of length 62, representing the character of that node, with an actual integer meaning an index exists at this particular branch, with the integer being the index. Searching through this tree is effectively simple, converting supplied id's and name's into values using the ordinance function as above, and then searching the array inside that point (returning nothing if no such array exists), until an integer is found which represents the index and is then added to an array of solutions. Due to the construction and search being a Trie, the following is true for this solution:

```
def query(filename, id_prefix, last_name_prefix):
    '''
    Description:        This function calls the queries through the data to find
all results that match id_prefix and
                        last_name_prefix.
    Time Complexity:    Best:   O(k + l + n(k), + n(i)), where k is the length of
the id, l is the length of the name,
                                n(k) is the number of records matching the
id_prefix and n(l) is the number of records
                                matching the last_name_prefix
                        Worst:  O(k + l + n(k), + n(i)), where k is the length of
the id, l is the length of the name,
                                n(k) is the number of records matching the
id_prefix and n(l) is the number of records
                                matching the last_name_prefix
    Space Complexity:   Best    O(N), where N is the size of solutions
                        Worst:  O(N), where N is the size of solutions
    Error Handle:       Not required, handled by caller
    Return:             Returns FinalSolutions
    Precondition:       It is given a filename, id_prefix and a last_name_prefix.
    '''
```

Task 2: Finding all palindromic substrings

The solution used in Task 2 was as follows, first, all substrings of the word reversed are created, which are then stored inside of an array. Following this creation, it is then checked (in linear time), whether the reversed substring in the array exists in the word, if it does, built in function find is used (time complexity O(1)) to find the position of the word, which is then appended to a solutions list, which is finally returned once all the words have been searched through. The following are the parameters, complexities and returns of the function:

```
def reverseSubstrings(filename):
    '''
    Description:        Finds all palindromic substrings of the first line of a
file.
    Time Complexity:    Best:   O(N^2 + P), where N is the number of characters in
the input string and P is the total
                        length of all substrings whose reverse appears in the
string
                        Worst:  O(N^2 + P), where N is the number of characters in
the input string and P is the total
                        length of all substrings whose reverse appears in the
string
    Space Complexity:   Best    O(N^2 + P), where N is the number of characters in
the input string and P is the total
                        length of all substrings whose reverse appears in the
string
                        Worst:  O(N^2 + P), where N is the number of characters in
the input string and P is the total
                        length of all substrings whose reverse appears in the
string
    Error Handle:       If a crash is caused, will name the file that caused the
crash for debugging
    Return:             Returns the 2D array of all palindromic substrings
    Precondition:       A filename is passed to the function
    '''
```