**Documentation**

SocialSim is a small implementation of the knowledge of programming, especially data structures and algorithms. This is simply a command line interface that implements the idea of social networking.

This project implements data structures like queues and graphs and also some Graph based searching algorithms like Breadth First Search and Depth First Search.

***Overview***

This project make use the ‘argparse’ package to implement and parse the arguments passed in the command-line.

Simply running SocialSim will tell you the usage and commands that can be used to run different modes of SocialSim.

There are two modes in which SocialSim can run. These are:

1. Interactive Testing Mode: We can enter this mode by simply running the command `SocialSim -i`. There are many options/choices that are given to you in this mode. We can do different types of operations in this mode like Loading network from some file, node operations like adding a person, finding a person and deleting a person, edge operations like liking / unliking a post, follow/unfollow a person etc, creating a new post, displaying and visualizing a network, saving a runtime network in a file and much more.

2. Simulation Mode: This mode allows a user to add a test folder in arguments and the program will automatically load, visualize/display the network.

***Implementation of Graph***

I have stored graphs as adjacency list.

Each edge of graph(network of SocialNetwork class) shows a follow relationship, where u -----> v represents that a person ‘u’ is following another person ‘v’.

I have chosen graph for storing the network because the data we want to store is making relation with each other like followers and following. This data is in a form of network. So, graph makes the implementation of networks much easier.

***Created Classes***

***1. SocialNetwork:*** The social network is a class which helps us store and manage names of people, network and posts of people.

‘names’ is simple a list that stores the name of the person on the unique index of the list. This corresponding index of this list is used as the unique id of the person in the graph(network) and list of list(posts).

‘network’ is simply an adjacency list implementation of a graph. Each node of this network consist of a list of index of persons he/she is following and this index is directly linked with the name of the person in ‘names’.

‘posts’ is a list of lists and each list on a specific index indicates the list of posts of the person associated with that index. Each post is an object of class ‘Post’.

Except the data variables of SocialNetwork class, it also contains many methods to perform different functionality of the class. Eg: findNode, createNode, deleteNode, addEdge, createPost, getNetwork, getPosts, breadthFirstSearch, depthFirstSearch etc.

‘findNode(u)’ is a method that takes a name as a parameter and return the index of the person with that name. If that name in the parameter is not found in the names list then, it returns -1.

‘createNode(u)’ is a method that takes a name of a person who want to make an account on SocialSim as parameter. This name is added at the end of the names list and the last index is then assigned to that person. Also, empty list of network and an empty list of posts is also created that shows that the person its not following anybody and the new person has posted nothing.

‘deleteNode(u)’ takes the name of the person as parameter and find the index of the person in the names list and deletes that element from all the data i.e, names, network and posts.

‘addEdge(u, v)’ takes the index of the person who want to follow some other person as ‘u’ and the index of the other person as ‘v’. This methods adds v to the following list of u in the network graph.

‘createPost(u, post)’ takes the index of the person who want to post and also an object of type Post which contains the title and content of the post. Obviously at the time of posting, there will be no likes the post.

‘getNetwork(u)’ takes the index of a person and returns a list of indexes of person whom the first person has followed.

‘getPosts(u)’ takes the index of a person and returns a list of all the objects of type Post that this user has posted.

‘breadthFirstSearch(start)’ takes the index of the starting point of the graph(network) and traverse each node that is linked with breadth order traversal.

‘depthFirstSearch(start)’ also takes the index of the starting point of the graph(network) and traverse each node that is linked with depth order traversal.

***2. Post:*** Post class defines the structure of a post posted by the user of the SocialSim network. Post contains three attributes:

‘title’ represents the title of the post.

‘content’ represents the content of the post.

‘likes’ is a list that stores the index of the persons who have liked the post.

***3. Queue:*** Queue class is used to implement the queue data structure used in breadthFirstSearch method of the graph.

Its contains methods like isEmpty, enqueue, dequeue, size. Their functions are same as the built-in data structure queue.