**Header file:**

#define MAX\_TWEET\_LEN 120

#define MAX\_HASH 1000

typedef struct Username

{

char username[10];// Username of twitter account; unique key

HEAP\* data;// Pointer to twitter data of user

}USERNAME;

typedef struct TwitterData

{

char date[10];// DateTime formatt: mm/dd/yyyy, hh:dd:mm:ss

int numberOfRetweets; // Used to check popularity of tweet

char tweet[180];// Tweet content

}TWITTERDATA;

int getData(BST \* usernameBST, USERNAME \* usernameHash[]);

void UIFunction(BST \* usernameBST, USERNAME usernameHash[]);

void newData(BST \* usernameBST, USERNAME usernameHash[]);

void deleteData(BST \* usernameBST, USERNAME usernameHash[]);

void displayHash(USERNAME usernameHash[]);

void displayBST(BST \* usernameBST);

void displayOne(USERNAME username[]);

void outputToFile(BST\* usernameBST);

void hash(char \*input)

void undoDelete(STACK\* undoStack);

void printEfficiency(USERNAME username[]);

void freeData(BST \* usernameBST, USERNAME usernameHash[]);

int cmpUsername(void\* name1, void\*name2);

int cmpDate(void\* date1, void\* date2);

int insertHash(USERNAME \* usernameHash[], char\* username)

**Rough Pseudocode (still in progress...):**

1. main
   1. createBST(cmpUsername)
   2. createStack()
   3. call getData()
      1. loop while not EOF
         1. read a line
         2. if insertBST() is not 0
            1. insertHash (usernameHash[], char\*username)

hash(char\*username)

* + - 1. else
         1. if username->heap is NULL

createHeap (cmpDate)

* + - * 1. insertHeap()
  1. UIFunction(ptr to hash, ptr to BST)
     1. newData // Adding new tweet to user
        1. scan username
        2. get system time
           1. parse the time into our string format
        3. get tweet (120 characters max)
        4. insertBST (if is not 0)
           1. insert in hash
        5. else
           1. if username->heap is not NULL

insertHeap()

* + - * 1. else

createHeap (cmpDate)

insertHeap()

* + 1. deleteData
       1. deleteBST
          1. hash //will return the index of wanted data to delete
    2. displayOne()
       1. index = hash()
       2. print index
    3. displayHash // loop through hashed array
    4. displayInOrder
       1. traverseBST
    5. displayBST
       1. traverseBST // (recursion)
    6. outputToFile
       1. inOrder
    7. printEfficiency
       1. // go through hash array
    8. undoDelete(STACK\*) //prompt user first if they want to undo then call this
       1. popstack(STACK\*,dataOut)
       2. displayOne(dataOut) and prompt if user wants to undo or not
       3. if (user wants to undo)
          1. insertBST and add to hash for dataOut
       4. else
          1. pushStack(STACK\*,dataOut)
    9. quit
  1. freeData
     1. loop userHash
        1. free heap
        2. free username
     2. destroyBST
     3. destroySTACK