Hacettepe University Computer Science Department
BBM203 Assignment 4

Name and Surname	: Ismet OKATAR
Identity Number	: 21727542
Course	: BBM203 Software Laboratory II
Experiment	: Assignment 4
Subject	: Login System with Character Tree
Date Due	: 06.01.2019
E-mail	: b21727542@cs.hacettepe.edu.tr

3. Software Design Notes
3.1 Description of the program

This is a simple login system which uses a special data structutre named "Trie". You can create read and delete from that system. The main aim is to access the data fast like google; and reduce the memory which is needed to store the data by making it dynamic. We have used in that software a new data structure type which is "Dynamic" Trie . So you don`t have to set nodes child size to 26 (the letter number in english). You can dynamically resize the child size.

Actually that provides language independent login system which supports all kind of languages and letters.

3.1.1 Problem

There must be the functions for creating and accessing the Dynamic Tree.
we have to handle ;
***I have copied and pasted some information because those defines best the problem*
-a command for append
it has to add username and password
** If the first character of the username is not referenced by the root node, the username will be added to the tree starting from the root node.*
** If the first n character of the username exists on the tree, a branch occurs on the nth node for the last characters*

**The node which is the last character of the username has to hold the password for the given username.*
** If there is a username same as the given username, the application will give an output that "reserved username".*

-s command for search

it has to search the tree and if find the username it has to return the password.

** If the first character of the username is not referenced by the root node the application will give an output that "no record".*
** If the first n character of the username exists on the tree, but the remainder is not, the application will give an output that "incorrect username".*
** If all characters of the username exist on the tree, but the last character has no password, the application will give an output that "not enough username".*
**If all characters of the username exist on the tree and the last character has its password, the application will give an output that "password xxx".*

-q command for query

it has to search the tree and if the username and the password matches it has to return the information about it.

** If the first character of the username is not referenced by the root node the application will give an output that "no record".*
** If the first n character of the username exists on the tree, but the remainder is not, the application will give an output that "incorrect username".*
** If all characters of the username exist on the tree, but the last character has no password, the application will give an output that "not enough username".*
** If all characters of the username exist on the tree, but the last character has a different password from the given, the application will give an output that "incorrect password".*

-d command for delete

it has to delete the existing username from the tree.

** If the first character of the username is not referenced by the root node the application will give an output that "no record".*
** If the first n character of the username exists on the tree, but the remainder is not, the application will give an output that "incorrect username".*
** If all characters of the username exist on the tree, but the last character has no password, the application will give an output that "not enough username".*
** If all characters of the username exist on the tree, and the last character has the password, the application will delete all nodes which are not connected to another username. Then it will give an output that "deletion is successful".*

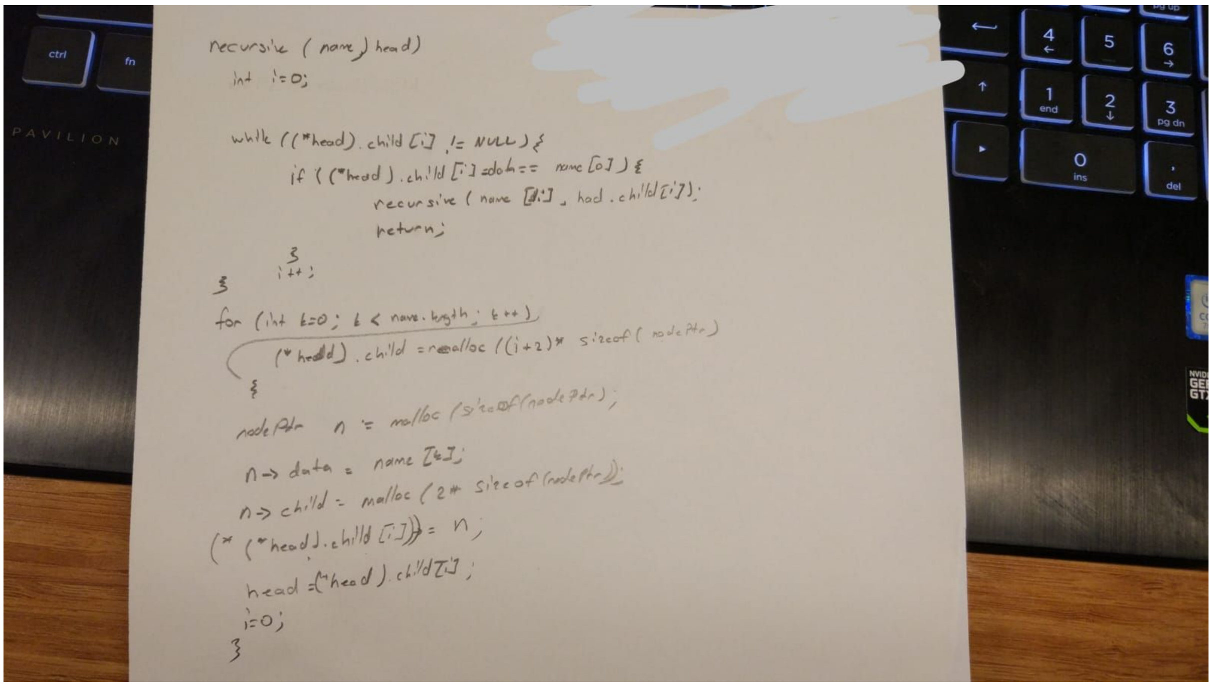
-l command for list.

it has to list all elements in the tree

3.1.2 Solution

The input.txt file must be readen for the data.
We have to declare a struct for our nodes.
Then we have to declare some functions for handling the given commands.
We have to implement a function for appending new users to the tree.

This is my first solving approach to that problem.



[redacted]

```
recursive ( name, head)
  int i=0;

  while ((*head).child [i] != NULL ){
       if (( (*head) .child [i] =data== name [0] ) {
              recursive ( name [1:] , had . child [i]);
              return;
       }
       i++;
}
for (int k=0; k < name.length ; k++ )
     (* head) . child = realloc ((i+2)* sizeof ( nodePtr)
  {
  nodePtr   n = malloc (sizeof(nodePtr));
  n-> data = name [k];
  n-> child = malloc (2* sizeof (nodePtr));
(* (*head).child [i])= n;
  head =(*head). child[i] ;
  i=0;
  }
```

*My first approach to the algorithm when i get the assignment.

After that algorithm i have modified and applied that logic to other things like listing deleting and searching some node from the tree.

3.2 Main Data Structures

i have used a Trie which is Dynamic. Struct type of my each node is below;
s

```
typedef struct node * node_ptr;
struct node{ // struct of nodes for each character
   char    data; // the character data
   char    * pass; // the password which is a string
   node_ptr *child; // a pointer to the inner linked lists head
   node * next;   // pointer to next linked list
}* nodePtr;
```

3.3 Algorithm

```
int main(int argc, char *argv[])

create the head of the tree
allocate 2 memory for his children.
open files
scan a string from file
if str == "-a"
          get other data by scanning and call __a() function
          print his name and call the addRecurive() function
          // addRecursive is a recursive function whichs value
     is
          // name , password ,and head
          if current name length is 0 and current head has no
     password add password to it and break.
          while head->child[i] == name[0] call addRecursive by
     passing child of head and names substring.
          if current name length is 0 and current head has
     password break.
          reallocate child memory +1;
          create and add new nodes to that head node.
     if last one assign also the password to it.
if str == "-s"
     get other data by scanning and call __s() function
          print his name and call the searhcRecurive() function
     by passing 's' char its mean it is search mode.
          // searchRecursive is a recursive function whichs
     value is
          // head, name, pass ,length ,mod
     if the conditions have reached print related outputs
     else
          keep moving around the tree
if str == "-q"
     do same thing with -s command but assign the mod value to
     'q'.
```

```
if str == “-d”
     get other data by scanning and call __s() function
          print his name and call the deleteRecurive()
     function.
          // deleteRecursive is a recursive function whichs
     value is
          // head, name, length, bulundu
     if found the last node free(head); set bulundu to 1.
     return;
     if the conditions have reached print other related
     outputs.
     else keep traveling in tree.
          ** because when we free a node it gonna be null my
     software would not work. Thats why i have inserted ‘0’ to
     freed positions to prevent it.
if str == “-l”
     get other data by scanning and call __l() function
          print his name and call the printRecurive() function.
          // printRecursive is a recursive function whichs
     value is
          // head, v , str
          while not reached to a node with password keep adding
     nodes to str array. if reached to password  print it on
     the screen.
```