There is some knowledge that related to basic binary tree function. And we’ll used these words for basic function in binary tree.

**add tree node function:** Get new node and root note of tree then add new node to binary treewhich compare with E-mail. If there are same E-mail, it should print error.

**search tree function:** Find node that want to know in binary treewhich find with E-mail. If found data, return address of user back. If not, return NULL.

**delete tree node function:** Get node that want to delete and root node of tree then find node that want to delete in tree and deleted it. If can delete, return 1, If can’t, return 0.

And some knowledge in Linked list and stack. We’ll used these words for basic function.

**find node in linked list**: Find node in linked list by we’ll send to find only head node of linked list

**delete node in linked list:** After find and found node want to delete, set another node that related to delete node to point correctly before delete. Then free node that want to delete.

**push data**: Add new data into head node of linked list

**pop data:** Get data of head node and free head node of linked list

**peek data:** Get data of head node but not free head node of linked list

**Add New User Data**

Ask data from user

Send E-mail to hash function

Get position of hash function

If there isn’t any data

Set that position in hash table to new data

End if

Else if there is some data

Get root node tree of that position in hash table

Send new data and root node tree to **add tree node** **function**

End if

**AVL Balancing tree**

After traversal in tree and **add tree node** **function**

**RECURSIVE FUNCTION**

Get height of left node and height of right node of current tree node

If height of left node is greater than height of right node

Set height of current tree node to one plus height of left node

End if

Else if height of right node is greater than height of left node

Set height of current tree node to one plus height of right node

End if

Check that height of left minus height of right equal to 1,0,-1 or not

If it’s not equal to 1,0,-1

Find out for type of unbalancing tree

If it’s unbalancing tree left of left

Rotate current node to the right

End if

If it’s unbalancing tree right of right

Rotate current node to the left

End if

If it’s unbalancing tree right of left

Rotate left node to the left

Rotate current node to the right

End if

If it’s unbalancing tree left of right

Rotate right node to the right

Rotate current node to the left

End if

Endif

Return sub tree after rotate to parent node

**Find User Data**

Set found user to NULL

Get E-mail from user

Send E-mail to hash function

Get position of hash function

If that position of hash table is NULL

Print not found message

Return found user

End if

Else

Get root node tree of that position in hash table

Send E-mail and root node tree to **search tree function**

Set found user to get data from search

If found user equal NULL

Print not found message

Return found user

End if

Else if found user not equal NULL

Print data profile

Return found user

End if

End if

**Send Pending Request**

Set current user to user that login

\*After find another user\*

Set find user to user that we find

Get Command from user

If user want to send pending request to be friend with find user

Send find user to **find node in linked list** **of friend** of current user

If found

Print to user they’re already friend

End if

Send find user to **find node in linked list** **of pending request** of current user

If found

Print to user that check in pending request first

End if

Send current user to **find node in linked list** **of pending request** of find user

If found

Print to user that user have been already send pending request

End if

Else

**Push current user** into linked list of pending request of find user

End if

End if

**See Friend Pending Request**

Set current user to user that login

Set request user to NULL

Get Command from user

If user want to see friend pending request

While friend pending request not equal to NULL

**Peek data** of linked list of head node of pending request

Print data

Ask command for user

If accept friend

**Pop data** of linked list of pending request keep in request user

**Push request user** into linked list of friend of current user

**Push current user** into linked list of friend of request user

End if

If deny friend

**Pop data** of linked list of pending request

End if

End while

End if

**Unfriend**

Set current user to user that login

\*After find another user\*

Set find user to user that we find

Get Command from user

If user want to unfriend with find user

Send find user to **find node in linked list** **of friend** of current user

If found

**delete find user in linked list of friend** of current user

**find current user in linked list of friend** of find user

**delete current user in linked list of friend** of find user

End if

Send find user to **find node in linked list** **of pending request** of current user

If found

Print to user that check in pending request first

End if

Send current user to **find node in linked list** **of pending request** of find user

If found

**delete current user in linked list of pending request** of find user

End if

Else

Print to user that they’re not friend

End if

End if

**Build profile file**

Set i to 0

Set max to maximum bucket of hash table

Open new profile file (binary file)

While i less than max

Get root node in i position of hash table

Post order traversal in tree

Write data into file

Free data

End post order traversal

Plus one to i

End while

**Build relation file**

Set i to 0

Set max to maximum bucket of hash table

Set friend node to NULL

Open new relation file (text file)

While i less than max

Get root node in i position of hash table

Post order traversal in tree

If friend of current node or pending request of current node are not NULL

Write down tag ‘USER’ into file

Write mail of current node into file

End if

Set friend node to head node of friend of user

While friend node not NULL

Write down tag ‘FRIEND’ into file

Write mail of friend node into file

Set friend node to next node

End while

Set friend node to head node of pending request of user

While friend node not NULL

Write down tag ‘PENDING’ into file

Write mail of friend node into file

Set friend node to next node

End while

End post order traversal

Plus one to i

End while

**Build status file**

Set i to 0

Set max to maximum bucket of hash table

Set status node to NULL

Set comment node to NULL

Open new relation file (text file)

While i less than max

Get root node in i position of hash table

Post order traversal in tree

Set status node to status of current node

While status node not NULL

Write down tag ‘STATUS’ into file

Write down text of status into file

Write down tag ‘MAIL’ into file

Write down mail of owner of status into file

Write down tag ‘DATE’ into file

Write down date of status into file

Set comment node to comment of status

while comment node not NULL

Write down tag ‘COMMENT’ into file

Write down text of comment into file

Write down tag ‘MAIL’ into file

Write down mail of owner of status into file

Write down tag ‘DATE’ into file

Write down date of status into file

Set comment node to next node

End while

Set status node to next node

End while

End post order traversal

Plus one to i

End while