<<AVLTree>>
Interface Data Type

<u>Owner:</u> Sam Warley

Struct Node:
- data:T = templated data
- left:Node*
- right:Node*
- level:int
- height:int

Class AVLTree

Public:

- AVLTree():constructor

- is_Empty():bool

- getHeight():int
- getLevel():int
- getTotalNodes():int

- getRootData():T
- setRootData(const T&) const:void

- insert(T& data) const:Node*
- remove(T& data) const:Node*
- search(T& data) const:bool

- printInOrder():void
- printPreOrder():void
- printPostOrder():void

Private:
- root:Node*
- dataQuantity:int

- countNodes([node* root]):int *;works with getTotalNodes()*

- bRotateWLeftChild(Node *temp):Node*
- bRotateWRightChild(Node *temp):Node*

- bBalanceCase2Left(Node *temp):Node*
- bBalanceCase3Right(Node *temp):Node*

<u>NOTE:</u> *Not sure if this will work with templates with balancing. Currently templated but in INT data type form*

<<QueryProcessor>>
Interface Data Processor

<u>Owner:</u> Sam Warley

Class QueryProcessor()

Public:

- QueryProcessor():constructor

- runSearch(String userInput):void *;for loop for*

- parseInput(String):void *;breaks down user input*

- returnSearchResults():vector<index>

- returnUserInput():String

Private:
- inquiryOrWords:vector<String>
- inquiryAndWords:vector<String>
- rawInquiryString:String
- andParse(String): void *;splits into words, inserts into and vector*
- orParse(String): void *;splits into words, inserts into or vector*
- notParse():void *;essentially a void function if NOT is called*

- outputUnsorted:vector<index>
- outputSorted:vector<index>

- sortSearch():void

<u>NOTE:</u> *Not sure what other implementation will be in index or document. Will be sorted based on the objects returned relevancy to the input*

<<UserInterface>>
Interface with User

Owner: Sam Warley

Class UserInterface()

Public:

- UserInterface():constructor
- userMenu():void *;while bool status = true*
- queryInput(string):void *;passes input to query processor*
- queryOutput():void *;calls runSearch*
- displaySearchResults():void *;calls returnSearchResults, formats and displays the search to the user*
- displaySearchResults(int reqSearchLoc):void *;calls returnSearchResults, formats and displays the search requested by int input to the user*
- displayCurrentSearch():void *;will display what was searched*
- listAllSearches():void *;allows users to view results for specific past searches*
- addNewSearch(string):void *;will add search to vector and run the search*
- accessSearch(int):void *;menu to access search requested*

Private:

- status:bool *;bool to exit search menu*
- lastSearch:int *;has the last search done, essentially a int for last entry to vector*
- searches:vector<QueryProcessor>

NOTE: Depending on the progress from other members of the team, I will attempt to implement a user GUI to help deal with the searches.