**MergeSort**

void Letter::orderWords( void ){

letterMergeSort(0,getWordCount()-1);

}

void Letter::letterMerge(int low, int middle, int high){

char\*\* temp = new char\*[high+1];

int\*\* tempInt = new int\*[high+1];

int\* tempIndexWidth = new int[high+1];

int kk;

int leftCounter = low;

int rightCounter = middle+1;

int tempCounter = low;

//put values into A array

while((leftCounter<=middle)&&(rightCounter<=high)){

//would need overloaded compare

if(compareString(wordList[leftCounter],wordList[rightCounter])){

temp[tempCounter] = wordList[leftCounter];

tempInt[tempCounter] = indexList[leftCounter];

tempIndexWidth[tempCounter] = indexWidthList[leftCounter];

leftCounter++;

}

else{

temp[tempCounter] = wordList[rightCounter];

tempInt[tempCounter] = indexList[rightCounter];

tempIndexWidth[tempCounter] = indexWidthList[rightCounter];

rightCounter++;

}

tempCounter++;

}

if(leftCounter>middle){

for(kk = rightCounter;kk<=high;kk++){

temp[tempCounter] = wordList[kk];

tempInt[tempCounter] = indexList[kk];

tempIndexWidth[tempCounter] = indexWidthList[kk];

tempCounter++;

}

}

else{

for(kk=leftCounter;kk<=middle;kk++){

temp[tempCounter] = wordList[kk];

tempInt[tempCounter] = indexList[kk];

tempIndexWidth[tempCounter] = indexWidthList[kk];

tempCounter++;

}

}

//reassign passed array to the now ordered elements in temp

for(int m = low;m<=high;m++){

wordList[m] = temp[m];

indexList[m] = tempInt[m];

indexWidthList[m] = tempIndexWidth[m];

}

//free memory

delete[] temp;

delete[] tempInt;

delete[] tempIndexWidth;

}

void Letter::letterMergeSort(int left, int right){

int middle;

if(left<right){

middle = (left+right)/2;

//continuatlly divides left side of A into smaller partitions

letterMergeSort(left,middle);

//continuatlly divides right side of A into smaller partitions

letterMergeSort(middle+1,right);

letterMerge(left,middle,right);

}

}

bool Letter::compareString(char\* firstWord, char\*secondWord){

//find string length of both word

int length1 = strlen(firstWord);

int length2 = strlen(secondWord);

int counter;

for(counter = 0;(counter<length1)&&(counter<length2);counter++){

if(toupper(firstWord[counter])<toupper(secondWord[counter]) ){

return true;

}

if(toupper(firstWord[counter])>toupper(secondWord[counter]) ){

return false;

}

}

//if the same word was encounterd

if(length1 == length2){

return true;

}

if((counter==length1) || (length1 == length2)){

return true;

}

return false;

}