//sorts each index of each vector alphabetically

for (int i = 0; i < classes.size(); i++)

{

cout << " in sort function" << endl;

sort(classes[i].begin(), classes[i].end());

cout << "done sorting" << endl;

}

//display vector index and elements in each vector index

for (int i = 0; i < classes.size(); i++)

{

cout << "vector" << i << endl;

for (int j = 0; j < classes[i].size(); j++)

{

//cout << "index" << j << endl;

cout << classes[i][j] << endl;

}

}

//push\_back values into a vector,

//then insert each vector into the vector classes

for (int i = 0; i < numClasses; i++)

{

vector<string> temp; // holds multiple vectors in each index

for (int j = 0; j < numWords; j++)

{

string word;

cout << "Give me " << j << " words: " << endl;

cin >> word;

temp.push\_back(word);

}

classes.push\_back(temp); //push individual vectors onto vector of vectors

}

//‘==’ operator overloading of class Student, which compares

//if the FinalGrade of the current student is equal to the

//FinalGrade of another student.

//If yes, return true; otherwise, return false.

bool operator==(const Student &x)

{

if (finalGrade == x.finalGrade)

{

return true;

}

else

return false;

}

for (int i = 0; i < numClasses; i++)

{

vector<string> index;

for (int j = 0; j < numWords; j++)

{

temp = word; //make copy of word

sort(temp.begin(), temp.end()); //alphebatize temp word

//check first case

if (classes[0].size() == 0)

{

cout << "inside" << endl;

index.push\_back(word);

testIndex.push\_back(temp);

}

//if word coming in is in index 0

else if (temp.size() == testClasses[i][0].size())

{

//if sorted word == first index of class

if (temp == testClasses[i][0])

{

//check if word is unique

if (temp == testClasses[i][0])

{

break;

}

//push word into index if comparisons are made

index.push\_back(word);

}

}

//if word coming is not in index

else

{

//make new vector index

classes.push\_back(newClass);

}

break;

}

}

//pushing first word into vector twice

//instances after three wont work

if (classes.size() == 0 && testClasses.size() == 0)

{

cout << " pushing first word into classes " << endl;

//Push first word into vector

index.push\_back(word);

classes.push\_back(index);

cout << " pushing first word into testClasses " << endl;

//push copy of first word into vector

testIndex.push\_back(tempWord);

testClasses.push\_back(testIndex);

}

for (int i = 0; i < testClasses.size() && i < classes.size(); i++)

{

cout << "inside for loop" << endl;

//if copy of word is equal to the size of first element in each class of the sorted vector

if (tempWord.size() == testClasses[i][0].size())

{

cout << "length is same" << endl;

//if copy of word is equal to the first element in each class of the sorted vector

if (tempWord == testClasses[i][0])

{

cout << "is anagram -- pushing word into vector" << endl;

classes[i].push\_back(word); //push word onto vector

}

else

{

cout << "pushing into new vector" << endl;

index.push\_back(word); //push word into new vector

classes.push\_back(index); //push vector onto classes

}

}

//if copy of word is not the same length, or anagram of existing vectors

else

{

cout << "pushing into new vector" << endl;

index.push\_back(word); //push word into new vector

classes.push\_back(index); //push vector onto classes

}

//ellenbogen working need to get uniques

void add(string word)

{

vector<string> index; //holds vector

static vector<string> testIndex; //holds temporary vector

string tempWord; //holds copy of word

tempWord = word; //set tempWord == to word

//sort copy of word lexographically

sort(tempWord.begin(), tempWord.end());

for (int i = 0; i < testIndex.size(); i++)

{

//check if anagram

if (tempWord == testIndex[i])

{

classes[i].push\_back(word); //push word onto classes

return; //jump out of function

}

}

testIndex.push\_back(tempWord); //push tempWord into testIndex

index.push\_back(word); //push word into index

classes.push\_back(index); //push index into classes

}