Filter students to those who are the same major and at least one semester ahead of the student we are looking for recommendations for

Find the x number of students who took the most similar course load and had similar reviews(if applicable) of said courses

Priority given to course, secondary ranking is the professor of said course

Determine which classes these students took most often in the semester that the searcher is going to be enrolling in classes

Sort these classes from high to low based on average review of the class from the x number of students

The recommended classes are the highest 5 classes in this sorted list of classes

```
Students[20000] umassStudents;
sameMajorStudents =
filter(umassStudents.isSameMajor());
olderStudents =
filter(sameMajorStudents.isOlder());
olderStudentsPlusSimilarity[olderStudents.l
ength][2]
for(int j = 0; j < olderStudents; j++){
    olderStudentsPlusSimilarity[j][1] =
olderStudents[j];
    olderStudentsPlusSimilarity[j][2] = 0;
        //1 point per similar class, 0.25
bonus for same professor
        //iterating through list of previously
taken classes of the student looking for
recommendations
        for(int j = 0; j < studentPrevClasses;
j++){

if(olderStudents.prevClasses.contains(stud
```

```
entPrevClasses[j]) && professor.isSame())
                    olderStudentsPlusSimilarity[j]
[2] = olderStudentsPlusSimilarity[j][2] +
1.25;
                else
if(olderStudents.prevClasses.contains(stud
entPrevClasses[j]))
                    olderStudentsPlusSimilarity[j]
[2] = olderStudentsPlusSimilarity[j][2] + 1;
          }
}
//sorts by previously found similarity
topTenMostSimilarOlderStudents =
olderStudentsPlusSimilarity.filter(highestTen
SimilarityScores);
List futureClassCount <futureClasses,
count>
for(student in
topTenMostSimilarOlderStudents){
     for(class in futureClasses){
          if(futureClassCount.contains(class)){
               futureClassCount[class]++
```

```
        else

futureClassCount.append(<Class, 1>)

recommendedClasses =
futureClassCount.sort(count)
return recommendedClasses[1 - 5];
```