Brighton ANCELIN
2017-12-5
ECE 2036
Final Project Part 2

# GTL Scheduler Documentation - v2

The approach I took to this assignment was to encapsulate data as objects wherever possible, and to ensure that relationships were readily obtainable from most any form of data. This makes my program great for modularity, but requires more memory than is strictly necessary and may be slower than optimized algorithms.

The program first prompts the user for file locations, and subsequently tokenizes these CSV files. To tokenize, in this sense, means to split the entire file into lines, and subsequently one-dimensional vectors of cell values as strings. After tokenization, cell values are encapsulated into Course objects and Student objects.

First, the Course objects are made, each with a unique, serial ID number. The Course class retains a static list of all unique Course objects.

Next, the Student objects are made. Each Student object contains member Course objects, which represent said student's course requests. These Course objects are obtained by equating course code strings to unique Course objects via a static Course class method. Student objects are retained with a local vector object. A course roster (StudentList object) is made for each course at this point as well, and a vector of these rosters is assembled as the “rosters” variable.

At this point, the program iterates through all student's and adds their course conflicts to a CourseConflictMap object. Each Student object has a method which returns all unique course conflicts possible for them, given their course selections. These course conflicts are represented with CoursePair objects, which contain two Course objects each. The CourseConflictMap object stores each Student object in a StudentList object, indexed by the conflict pair the Student has. This means identical Student objects appear multiple times throughout the CourseConflictMap mapping.

The CourseConflictMap stores StudentList objects internally in an upper triangular matrix. The width of this square semi-matrix is one less than the number of courses available.

The aforesaid CourseConflictMap is copied into a new variable called “conflictsProfs”. All possible course pairings are iterated through and a dummy Student object is added as a conflict at any instance where the pairing shares a professor.

All courses are iterated through, giving the iterated Course object the name “c”. For each course, all timeslots (8AM, 9AM, etc.) are iterated through. For each timeslot, if there are less than 6 courses and none of the courses currently set at that time conflict with the “c” Course object (by “conflictsProfs” conflict definition, meaning there is a conflict if either a student or professor has or teaches both courses, respectively), add “c” to the timeslot. Once “c” is added to a timeslot, nested iteration of the timeslots will cease. If “c” never finds an existing timeslot to be inserted into, “c” is placed into its own timeslot appended at the end of the current “timeslots” vector.

Course conflicts are now written in the specified matrix format by retrieving the size of

StudentList objects mapped by the CourseConflictMap object.

Course conflicts detailing individuals data is also output in the specified format by reading the Student objects stored within the StudentList objects mapped by the CourseConflictMap object.

A schedule is output in the specified format by iterating through the “timeslots” vector and outputting the courses allocated in each timeslot.

Class rolls are output in the specified format by iterating through the “rosters” vector and outputting all students stored in each roster under the roster's Coruse object's course code.

**My computer/program processed everything for the long files in less than a second; the duration was unnoticeable for every test file.**

Console:

```
Please input Classes file name:
2017ClassesShortTest.csv
Please input Students file name:
2017StudentsShortTest.csv
Student "McDermitt, Tom, Junior" requested course "ISyE 3025", which is not
offered. Ignoring request.
Student "Smith, Allison, Wilson" requested course "ISyE 3025", which is not
offered. Ignoring request.
```

2017ClassesShortTest.csv

```
Course number,Course Title,Faculty
HTS 2084,Technology & Society,J. Krige
ECON 2101,Global Economics,C. Sonntag
MSE 2001,Principles and Applications of Engineering Materials,B. Carter
ChBE 3200,Transport Processes I,J. Schork
FREN 1002,Elementary French II,JC Ippolito
ISyE 3039,Methods for Quality Improvement,P. Kvam
ECE 3025,Electromagnetics,P. Voss
```

2017StudentsShortTest.csv

```
GTID,NAME,Course 1,Course 2,Course 3,Course 4
5,"McDermitt, Tom, Junior",HTS 2084,ECON 2101,MSE 2001,ISyE 3025
27,"Del Vincent, Julie, Laure",ChBE 3200,MSE 2001,ECON 2101,HTS 2084
115,"Smith, Allison, Wilson",HTS 2084,ChBE 3200,ECE 3025,ISyE 3025
127,"Brown, Thomas, Klett",ChBE 3200,ECON 2101,ECE 3025,HTS 2084
188,"Jones, Matthew Charles",ECON 2101,MSE 2001,ISyE 3039,FREN 1002
```

Short_theConflictMatrix.csv

```
,HTS 2084,ECON 2101,MSE 2001,ChBE 3200,FREN 1002,ISyE 3039,ECE 3025
HTS 2084,0,3,2,3,0,0,2
ECON 2101,3,0,3,2,1,1,1
MSE 2001,2,3,0,1,1,1,0
ChBE 3200,3,2,1,0,0,0,2
FREN 1002,0,1,1,0,0,1,0
ISyE 3039,0,1,1,0,1,0,0
ECE 3025,2,1,0,2,0,0,0
```

Short_whoHasConflicts.csv

```
List of class pairs and students with conflicts for that class pair:
For the course pair: HTS 2084 : ECON 2101
McDermitt, Tom, Junior
Del Vincent, Julie, Laure
Brown, Thomas, Klett

For the course pair: HTS 2084 : MSE 2001
McDermitt, Tom, Junior
Del Vincent, Julie, Laure

For the course pair: HTS 2084 : ChBE 3200
```

```
Del Vincent, Julie, Laure
Smith, Allison, Wilson
Brown, Thomas, Klett

For the course pair: HTS 2084 : FREN 1002

For the course pair: HTS 2084 : ISyE 3039

For the course pair: HTS 2084 : ECE 3025
Smith, Allison, Wilson
Brown, Thomas, Klett
…
```

Short_Schedule.txt

```
8:00 HTS 2084 FREN 1002
9:00 ECON 2101
10:00 MSE 2001 ECE 3025
11:00 ChBE 3200 ISyE 3039
```

Short_ClassRolls.csv

```
HTS 2084:
"McDermitt, Tom, Junior"
"Del Vincent, Julie, Laure"
"Smith, Allison, Wilson"
"Brown, Thomas, Klett"

ECON 2101:
"McDermitt, Tom, Junior"
"Del Vincent, Julie, Laure"
"Brown, Thomas, Klett"
"Jones, Matthew Charles"

MSE 2001:
"McDermitt, Tom, Junior"
"Del Vincent, Julie, Laure"
"Jones, Matthew Charles"

ChBE 3200:
"Del Vincent, Julie, Laure"
"Smith, Allison, Wilson"
"Brown, Thomas, Klett"
...
```

Console:

```
Please input Classes file name:
2017ClassesLongTest.csv
Please input Students file name:
2017StudentsLongTest.csv
```

2017ClassesLongTest.csv

```
Course number,Course Title,Faculty
ACCT 2101,Accounting I,A. Schneider
ChBE 3110,ChE Thermodynamics II,J. Schork
ChBE 3200,Transport Processes I,J. Schork
COE 2001,Statics,C. Valle
COE 3001,Deformable Bodies,C. Valle
CS 4400,Introduction to Database Systems,S. Rugaber
CS 4803,Media Based Software Engineering,S. Rugaber
ECE 2025,Intro Signal Processing,M. Bloch
ECE 2030,Intro to Computer Engineering,H. Owen
ECE 2040,Circuit Analysis,B. Boussert
ECE 3025,Electromagnetics,P. Voss
ECE 3040,Microelectronic Circuits,D. Citrin
ECE 3710,Circuits and Electronics,D. Citrin
ECON 2101,Global Economics,C. Sonntag
FREN 1001,Elementary French I,JC Ippolito
FREN 1002,Elementary French II,JC Ippolito
FREN 2813,Intermediate French Proficiency,JC Ippolito
FREN 3013,Introduction to Contemporary France,JC Ippolito
HTS 2084,Technology & Society,J. Krige
HTS 3064,Sociology of Development,T. Stoneman
HTS 4823,History of Rocketry,J. Krige
ISyE 2028,Basic statistical Methods,P. Kvam
ISyE 3025,Essentials of Engineering Economy,J. Sands
ISyE 3039,Methods for Quality Improvement,P. Kvam
ISyE 3770,Statistics and Applications,A. Locquet
MATH 2401,Calculus III,H. Matzinger
MATH 2403,Differential Equations,H. Matzinger
ME 3322,Thermodynamics,N. Declercq
MGT 3660,International Business,S. D'Eyrames
MSE 2001,Principles and Applications of Engineering Materials,B. Carter
```

2017StudentsLongTest.csv

```
GTID,NAME,Course 1,Course 2,Course 3,Course 4
1,"Armhand, Anant Shirish",ISyE 3025,FREN 1001,MATH 2401,ECE 2030
2,"Arq, Helen ",ECE 3710,MATH 2403,ISyE 3770,COE 3001
3,"Bach, Emilia G",CS 4400,HTS 4823,ISyE 3770,
4,"Baki, Zainab, Saifudin",MATH 2401,FREN 1001,MATH 2403,ECE 2040
5,"Banfar, Marc Richard",ChBE 3110,ChBE 3200,MSE 2001,
6,"Barnes, Shashank, Sundar",ECE 2025,MATH 2401,HTS 2084,
7,"Barren, Ryan, James",ChBE 3110,ChBE 3200,HTS 2084,
8,"Barthelot, Virgille, Malapit",ChBE 3110,ChBE 3200,ISyE 2028,
9,"Bes Cab, Sukirat Singh",ISyE 3770,MSE 2001,HTS 4823,
10,"Bog, Drew, Jordan",ISyE 2028,ISyE 3025,FREN 1002,CS 4400
11,"Borzen, Dimple, Pushpendra",CS 4400,ISyE 2028,ACCT 2101,ECE 2030
12,"Bourges, Andres",ISyE 3039,ISyE 3025,ECE 2030,MSE 2001
13,"Brevt, Andrew Ryan",ECE 2025,HTS 2084,FREN 1001,
```

```
14,"Buss, Kyle, Zachary",ISyE 3770,ECE 2025,ECON 2101,COE 2001
15,"Cal, Andres Felipe",ISyE 3039,ISyE 3025,CS 4400,FREN 1001
16,"Carns, William, Daniel",MATH 2401,ECON 2101,MSE 2001,
17,"Cast, Lane Austin",ISyE 2028,MSE 2001,CS 4400,
18,"Chaw, Patrick, Abell",COE 3001,ME 3322,ISyE 3770,ISyE 3025
19,"Cheeva, Michael, Eugene",ISyE 3025,ISyE 3039,ME 3322,FREN 1001
20,"Cheng, Maricruz",COE 3001,ME 3322,HTS 2084,
21,"Chew, Jorge, Luis",ChBE 3110,ChBE 3200,FREN 1001,
22,"Chile, Betsy, Erin",FREN 1002,MATH 2401,ECE 2040,
23,"Chop, Kelly, Ann",ISyE 2028,ACCT 2101,CS 4400,
24,"Cleme, Hannah, Ray",ISyE 2028,MATH 2401,FREN 1001,
25,"Cona, Angad, Singh",COE 3001,ME 3322,ISyE 3770,HTS 2084
26,"Corbed, Nitid",ME 3322,ISyE 3770,ECON 2101,
27,"Crowert, Nakul, Devang",MATH 2401,ACCT 2101,MSE 2001,
28,"Dabri, Stanley, Chunwing",ECE 3710,ISyE 3025,HTS 2084,HTS 3064
29,"Daved, Bethany, Maria ",ECE 2030,ISyE 3025,ACCT 2101,ISyE 3039
30,"Del Gebert, Andrew, James",ISyE 3039,ECE 2030,ISyE 3025,ECE 3710
...
```

Long_theConflictMatrix.csv

```
,ACCT 2101,ChBE 3110,ChBE 3200,COE 2001,COE 3001,CS 4400,CS 4803,ECE 2025,ECE
2030,ECE 2040,ECE 3025,ECE 3040,ECE 3710,ECON 2101,FREN 1001,FREN 1002,FREN
2813,FREN 3013,HTS 2084,HTS 3064,HTS 4823,ISyE 2028,ISyE 3025,ISyE 3039,ISyE
3770,MATH 2401,MATH 2403,ME 3322,MGT 3660,MSE 2001
ACCT 2101,0,1,0,0,2,7,0,0,2,0,1,1,0,2,6,3,0,0,0,0,3,4,6,4,0,3,1,1,5,3
ChBE 3110,1,0,17,0,0,0,0,0,0,0,0,0,0,2,4,2,0,0,4,1,0,1,0,0,0,0,2,0,1,9
ChBE 3200,0,17,0,0,0,0,0,0,0,0,0,0,0,2,3,2,0,0,4,1,1,1,0,0,0,0,2,0,0,10
COE 2001,0,0,0,0,0,0,4,0,9,2,1,0,0,7,6,6,2,4,1,5,0,3,2,5,7,10,4,15,3,2,13
COE 3001,2,0,0,0,0,0,1,0,0,0,0,0,2,0,5,0,4,3,1,0,9,0,4,0,6,0,9,2,7,8,0,5
CS 4400,7,0,0,4,1,0,0,1,4,0,0,0,0,4,5,7,0,4,3,0,2,11,23,15,2,3,0,0,3,6
CS 4803,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
ECE 2025,0,0,0,9,0,1,0,0,1,0,0,0,1,2,7,2,0,0,7,0,0,0,1,1,9,3,8,0,1,7
ECE 2030,2,0,0,2,0,4,0,1,0,0,0,0,2,2,2,0,0,1,2,0,0,2,8,7,1,1,0,0,0,1
ECE 2040,0,0,0,1,0,0,0,0,0,0,1,0,0,1,1,1,0,0,1,0,0,0,0,0,1,2,3,0,0,0
ECE 3025,1,0,0,0,2,0,0,0,0,1,0,8,2,1,2,0,1,0,2,0,2,1,2,1,3,0,1,4,0,1
ECE 3040,1,0,0,0,0,0,0,0,0,0,8,0,1,1,1,0,0,0,0,0,2,0,0,0,2,0,0,3,0,0
ECE 3710,0,0,0,7,5,0,0,1,2,0,2,1,0,1,2,1,0,0,4,1,3,0,5,1,9,0,5,6,0,2
ECON 2101,2,2,2,6,0,4,0,2,2,1,1,1,1,0,5,4,4,2,15,3,1,4,6,3,4,6,4,2,3,8
FREN 1001,6,4,3,6,4,5,0,7,2,1,2,1,2,5,0,1,0,0,8,2,7,3,13,6,8,7,5,6,7,7
FREN 1002,3,2,2,2,3,7,0,2,0,1,0,0,1,4,1,0,0,0,11,3,1,1,5,2,5,4,2,5,1,5
FREN 2813,0,0,0,4,1,0,0,0,0,0,1,0,0,4,0,0,0,1,2,0,2,2,1,1,0,1,1,2,2,2
FREN 3013,0,0,0,1,0,4,0,0,1,0,0,0,0,2,0,0,1,0,1,1,1,1,2,2,1,0,0,0,1,1
HTS 2084,0,4,4,5,9,3,0,7,2,1,2,0,4,15,8,11,2,1,0,6,5,1,8,2,15,5,5,8,3,11
HTS 3064,0,1,1,0,0,0,0,0,0,0,0,0,1,3,2,3,0,1,6,0,2,0,1,0,0,0,0,0,0,0
HTS 4823,3,0,1,3,4,2,0,0,0,0,2,2,3,1,7,1,2,1,5,2,0,0,2,0,2,1,2,0,2,6
ISyE 2028,4,1,1,2,0,11,0,0,2,0,1,0,0,4,3,1,2,1,1,0,0,0,10,3,0,6,1,0,0,2
ISyE 3025,6,0,0,5,6,23,0,1,8,0,2,0,5,6,13,5,1,2,8,1,2,10,0,19,8,8,1,7,2,6
ISyE 3039,4,0,0,7,0,15,0,1,7,0,1,0,1,3,6,2,1,2,2,0,0,3,19,0,0,1,1,1,4,5
ISyE 3770,0,0,0,10,9,2,0,9,1,1,3,2,9,4,8,5,0,1,15,0,2,0,8,0,0,0,8,18,0,6
MATH 2401,3,0,0,4,2,3,0,3,1,2,0,0,0,6,7,4,1,0,5,0,1,6,8,1,0,0,4,0,0,6
MATH 2403,1,2,2,15,7,0,0,8,0,3,1,0,5,4,5,2,1,0,5,0,2,1,1,1,8,4,0,0,1,17
ME 3322,1,0,0,3,8,0,0,0,0,0,4,3,6,2,6,5,2,0,8,0,0,0,7,1,18,0,0,0,1,1
MGT 3660,5,1,0,2,0,3,0,1,0,0,0,0,0,3,7,1,2,1,3,0,2,0,2,4,0,0,1,1,0,3
MSE 2001,3,9,10,13,5,6,0,7,1,0,1,0,2,8,7,5,2,1,11,0,6,2,6,5,6,6,17,1,3,0
```

Long_whoHasConflicts.csv

```
List of class pairs and students with conflicts for that class pair:
For the course pair: ACCT 2101 : ChBE 3110
Srini, Nul, Sam

For the course pair: ACCT 2101 : ChBE 3200

For the course pair: ACCT 2101 : COE 2001

For the course pair: ACCT 2101 : COE 3001
Mikel, Reid, James
Pateler, Courtney, Elizabeth

For the course pair: ACCT 2101 : CS 4400
Borzen, Dimple, Pushpendra
Chop, Kelly, Ann
Grandad, Aakash, Harish
Hollow, Kelly, Lynn
Kinner, Hrishikesh, Sanjeev
Sent, Ryan, Michael
Yiha, Andrew, Phillip

For the course pair: ACCT 2101 : CS 4803

For the course pair: ACCT 2101 : ECE 2025

For the course pair: ACCT 2101 : ECE 2030
Borzen, Dimple, Pushpendra
Daved, Bethany, Maria

For the course pair: ACCT 2101 : ECE 2040

For the course pair: ACCT 2101 : ECE 3025
Schweber, Andrew, Keishi

For the course pair: ACCT 2101 : ECE 3040
Schweber, Andrew, Keishi

For the course pair: ACCT 2101 : ECE 3710

For the course pair: ACCT 2101 : ECON 2101
Hollow, Kelly, Lynn
Yiha, Andrew, Phillip

For the course pair: ACCT 2101 : FREN 1001
Engel, Dhrumil, Mitesh
Leesmith, Connor, Anthony
Mikel, Reid, James
Pear, Ryan, Teddy
Scheber, Abdullah Omar
Srini, Nul, Sam
…
```

Long_Schedule.txt

```
8:00 ACCT 2101 ChBE 3200 COE 2001 CS 4803
9:00 ChBE 3110 COE 3001 ECE 2025 ECE 2040 ECE 3040 FREN 3013
10:00 CS 4400 ECE 3025 HTS 3064
11:00 ECE 2030 FREN 1002
12:00 ECE 3710 FREN 2813
```

13:00 ECON 2101
14:00 FREN 1001
15:00 HTS 2084
16:00 HTS 4823 ISyE 2028 ME 3322
17:00 ISyE 3025
18:00 ISyE 3039 ISyE 3770
19:00 MATH 2401 MGT 3660
20:00 MATH 2403
21:00 MSE 2001

## Long_ClassRolls.csv

ACCT 2101:
"Borzen, Dimple, Pushpendra"
"Chop, Kelly, Ann"
"Crowert, Nakul, Devang"
"Daved, Bethany, Maria "
"Engel, Dhrumil, Mitesh"
"Grandad, Aakash, Harish"
"Han, Carolyn, Anne"
"Hollow, Kelly, Lynn"
"Kinner, Hrishikesh, Sanjeev"
"Leesmith, Connor, Anthony"
"Mikel, Reid, James"
"Pateler, Courtney, Elizabeth"
"Pear, Ryan, Teddy"
"Qamru, Sarah, Elizabeth"
"Scheber, Abdullah Omar"
"Schweber, Andrew, Keishi"
"Sent, Ryan, Michael"
"Shell, Andrew, Christopher"
"Srini, Nul, Sam"
"Yanbert, Alexander, Tzejung"
"Yiha, Andrew, Phillip"

ChBE 3110:
"Banfar, Marc Richard"
"Barren, Ryan, James"
"Barthelot, Virgille, Malapit"
"Chew, Jorge, Luis"
"Depp, Nikhil"
"Fech, Anna, Viktoria"
"Ferret, Benjamin Evan"
"Gerenomo, Christine, Diane"
"Girar, Zechuan"
"Haug, Sahil"
"Hoxer, Alejandro, Rafael"
"LaPass, Walter, Spencer"
"Lewert, Putchararporn, Paula"
"Nadir, Dante, Edward"
"Neo, Lishan"
"Shert, Christopher, Bradley"
"Srini, Nul, Sam"
"Western, Sarvagya"
"Zarn, Justin Russell"

…

Console:

```
Please input Classes file name:
2017_1C.csv
Please input Students file name:
2017_1S.csv
File "2017_1S.csv" has an incomplete line. This line will be ignored.
```

2017_1C.csv

```
Course number,Course Title,Faculty
MUS 2000,Music in Industry,W. A. Mozart
HUM 1342,Society in the Humanities,Q. Rius
MSE 2001,Principles of Engineering Materials,T. Sanders
INTA 2200,International Affairs in Europe,C. Claire
ECON 4001,Senior Economics,A. Carnegie
HTS 3080,History of Rocketry,J. Krige
ECE 3072,Power Systems,N. Tesla
AE 9001,Special Topics: Get to Mars,TBD
```

2017_1S.csv

```
GTID,NAME,Course 1,Course 2,Course 3,Course 4
5141341,"Gomez, Selena",MUS 2000,HUM 1342,MSE 2001,INTA 2200
27323442,"Musk, Elon",ECON 4001,HTS 3080,ECE 3072,AE 9001
```

1_theConflictMatrix.csv

```
,MUS 2000,HUM 1342,MSE 2001,INTA 2200,ECON 4001,HTS 3080,ECE 3072,AE 9001
MUS 2000,0,1,1,1,0,0,0,0
HUM 1342,1,0,1,1,0,0,0,0
MSE 2001,1,1,0,1,0,0,0,0
INTA 2200,1,1,1,0,0,0,0,0
ECON 4001,0,0,0,0,0,1,1,1
HTS 3080,0,0,0,0,1,0,1,1
ECE 3072,0,0,0,0,1,1,0,1
AE 9001,0,0,0,0,1,1,1,0
```

1_whoHasConflicts.csv

```
List of class pairs and students with conflicts for that class pair:
For the course pair: MUS 2000 : HUM 1342
Gomez, Selena

For the course pair: MUS 2000 : MSE 2001
Gomez, Selena

For the course pair: MUS 2000 : INTA 2200
Gomez, Selena

For the course pair: MUS 2000 : ECON 4001

For the course pair: MUS 2000 : HTS 3080

For the course pair: MUS 2000 : ECE 3072
```

…

## 1_Schedule.txt

```
8:00 MUS 2000 ECON 4001
9:00 HUM 1342 HTS 3080
10:00 MSE 2001 ECE 3072
11:00 INTA 2200 AE 9001
```

## 1_ClassRolls.csv

```
MUS 2000:
"Gomez, Selena"

HUM 1342:
"Gomez, Selena"

MSE 2001:
"Gomez, Selena"

INTA 2200:
"Gomez, Selena"

ECON 4001:
"Musk, Elon"

HTS 3080:
"Musk, Elon"

ECE 3072:
"Musk, Elon"

AE 9001:
"Musk, Elon"
```

Console:

```
Please input Classes file name:
2017_2C.csv
Please input Students file name:
2017_2S.csv
```

2017_2C.csv

```
Course number,Course Title,Faculty
HTS 2084,Technology & Society,J. Krige
ECON 2101,Global Economics,C. Sonntag
MSE 2001,Principles and Applications of Engineering Materials,B. Carter
ChBE 3200,Transport Processes I,J. Schork
FREN 1002,Elementary French II,JC Ippolito
ISyE 3039,Methods for Quality Improvement,P. Kvam
CELB 2200,How to be Famous,S. LaBeouf
```

2017_2S.csv

```
GTID,NAME,Course 1,Course 2,Course 3,Course 4
5,"McDermitt, Tom, Junior",HTS 2084,ECON 2101,MSE 2001,ISyE 3039
27,"Del Vincent, Julie, Laure",ChBE 3200,MSE 2001,ECON 2101,HTS 2084
188,"Jones, Matthew Charles",ECON 2101,MSE 2001,ISyE 3039,FREN 1002
200,"Olsen, Kate",ECON 2101,ChBE 3200,ISyE 3039,CELB 2200
201,"Olsen, Ashley",ECON 2101,ChBE 3200,ISyE 3039,CELB 2200
```

2_theConflictMatrix.csv

```
,HTS 2084,ECON 2101,MSE 2001,ChBE 3200,FREN 1002,ISyE 3039,CELB 2200
HTS 2084,0,2,2,1,0,1,0
ECON 2101,2,0,3,3,1,4,2
MSE 2001,2,3,0,1,1,2,0
ChBE 3200,1,3,1,0,0,2,2
FREN 1002,0,1,1,0,0,1,0
ISyE 3039,1,4,2,2,1,0,2
CELB 2200,0,2,0,2,0,2,0
```

2_whoHasConflicts.csv

```
List of class pairs and students with conflicts for that class pair:
For the course pair: HTS 2084 : ECON 2101
McDermitt, Tom, Junior
Del Vincent, Julie, Laure

For the course pair: HTS 2084 : MSE 2001
McDermitt, Tom, Junior
Del Vincent, Julie, Laure

For the course pair: HTS 2084 : ChBE 3200
Del Vincent, Julie, Laure

For the course pair: HTS 2084 : FREN 1002

For the course pair: HTS 2084 : ISyE 3039
```

```
McDermitt, Tom, Junior

For the course pair: HTS 2084 : CELB 2200
...
```

## 2_Schedule.txt

```
8:00 HTS 2084 FREN 1002 CELB 2200
9:00 ECON 2101
10:00 MSE 2001
11:00 ChBE 3200
12:00 ISyE 3039
```

## 2_ClassRolls.csv

```
HTS 2084:
"McDermitt, Tom, Junior"
"Del Vincent, Julie, Laure"

ECON 2101:
"McDermitt, Tom, Junior"
"Del Vincent, Julie, Laure"
"Jones, Matthew Charles"
"Olsen, Kate"
"Olsen, Ashley"

MSE 2001:
"McDermitt, Tom, Junior"
"Del Vincent, Julie, Laure"
"Jones, Matthew Charles"

ChBE 3200:
"Del Vincent, Julie, Laure"
"Olsen, Kate"
"Olsen, Ashley"
…
```

Console:

```
Please input Classes file name:
2017_3C.csv
Please input Students file name:
2017_3S.csv
Duplicate course listing "MUS 2000". Ignoring the latter duplicate.
```

2017_3C.csv

```
Course number,Course Title,Faculty
MUS 2000,Music in Industry,W. A. Mozart
HUM 1342,Society in the Humanities,Q. Rius
MUS 2000,Duplicate Music Course,W. A. Zomart
```

2017_3S.csv

```
GTID,NAME,Course 1,Course 2,Course 3,Course 4
5141341,"Gomez, Selena",MUS 2000,HUM 1342
27323442,"Smith, John",HUM 1342,
```

3_theConflictMatrix.csv

```
,MUS 2000,HUM 1342
MUS 2000,0,1
HUM 1342,1,0
```

3_whoHasConflicts.txt

```
List of class pairs and students with conflicts for that class pair:
For the course pair: MUS 2000 : HUM 1342
Gomez, Selena
```

3_Schedule.txt

```
8:00 MUS 2000
9:00 HUM 1342
```

3_ClassRolls.csv

```
MUS 2000:
"Gomez, Selena"

HUM 1342:
"Gomez, Selena"
"Smith, John"
```

Console:

```
Please input Classes file name:
2017_4C.csv
Please input Students file name:
2017_4S.csv
File "2017_4C.csv" has an incomplete line. This line will be ignored.
Student "Musk, Elon" requested course "ECON 4001", which is not offered. Ignoring
request.
File "2017_4S.csv" has an incomplete line. This line will be ignored.
```

2017_4C.csv

```
Course number,Course Title,Faculty
MUS 2000,Music in Industry,W. A. Mozart
HUM 1342,Society in the Humanities,Q. Rius
MSE 2001,Principles of Engineering Materials,T. Sanders
INTA 2200,International Affairs in Europe,C. Claire
ECON 4001,Senior Economics
HTS 3080,History of Rocketry,J. Krige
ECE 3072,Power Systems,N. Tesla
AE 9001,Special Topics: Get to Mars,TBD
```

2017_4S.csv

```
GTID,NAME,Course 1,Course 2,Course 3,Course 4
5141341,"Gomez, Selena",MUS 2000,HUM 1342,MSE 2001,INTA 2200
27323442,"Musk, Elon",ECON 4001,HTS 3080,ECE 3072,AE 9001,MUS 2000
```

4_theConflictMatrix.csv

```
,MUS 2000,HUM 1342,MSE 2001,INTA 2200,HTS 3080,ECE 3072,AE 9001
MUS 2000,0,1,1,1,1,1,1
HUM 1342,1,0,1,1,0,0,0
MSE 2001,1,1,0,1,0,0,0
INTA 2200,1,1,1,0,0,0,0
HTS 3080,1,0,0,0,0,1,1
ECE 3072,1,0,0,0,1,0,1
AE 9001,1,0,0,0,1,1,0
```

4_whoHasConflicts.txt

```
List of class pairs and students with conflicts for that class pair:
For the course pair: MUS 2000 : HUM 1342
Gomez, Selena

For the course pair: MUS 2000 : MSE 2001
Gomez, Selena

For the course pair: MUS 2000 : INTA 2200
Gomez, Selena

For the course pair: MUS 2000 : HTS 3080
Musk, Elon
```

```
For the course pair: MUS 2000 : ECE 3072
Musk, Elon

For the course pair: MUS 2000 : AE 9001
Musk, Elon

For the course pair: HUM 1342 : MSE 2001
Gomez, Selena

For the course pair: HUM 1342 : INTA 2200
Gomez, Selena

For the course pair: HUM 1342 : HTS 3080

For the course pair: HUM 1342 : ECE 3072
…
```

4_Schedule.txt

```
8:00 MUS 2000
9:00 HUM 1342 HTS 3080
10:00 MSE 2001 ECE 3072
11:00 INTA 2200 AE 9001
```

4_ClassRolls.csv

```
MUS 2000:
"Gomez, Selena"
"Musk, Elon"

HUM 1342:
"Gomez, Selena"

MSE 2001:
"Gomez, Selena"

INTA 2200:
"Gomez, Selena"

HTS 3080:
"Musk, Elon"

ECE 3072:
"Musk, Elon"

AE 9001:
"Musk, Elon"
```

Console:

```
Please input Classes file name:
2017_5C.csv
Please input Students file name:
2017_5S.csv
File "2017_5S.csv" has an incomplete line. This line will be ignored.
```

2017_5C.csv

```
Course number,Course Title,Faculty
HTS 2084,Technology & Society,J. Krige
ECON 2101,Global Economics,C. Sonntag
MSE 2001,Principles and Applications of Engineering Materials,B. Carter
ChBE 3200,Transport Processes I,J. Schork
FREN 1002,Elementary French II,JC Ippolito
ISyE 3039,Methods for Quality Improvement,P. Kvam
CELB 2200,How to be Famous,S. LaBeouf
```

2017_5S.csv

```
GTID,NAME,Course 1,Course 2,Course 3,Course 4
5,"McDermitt, Tom, Junior",HTS 2084,ECON 2101,MSE 2001,ISyE 3039
27,"Del Vincent, Julie, Laure",ChBE 3200,MSE 2001,ECON 2101,HTS 2084
188,"Jones, Matthew Charles",ECON 2101,MSE 2001,ISyE 3039,FREN 1002
200,"Olsen, Kate",ECON 2101,ChBE 3200,ISyE 3039,CELB 2200
201,"Olsen, Ashley"
```

5_theConflictMatrix.csv

```
,HTS 2084,ECON 2101,MSE 2001,ChBE 3200,FREN 1002,ISyE 3039,CELB 2200
HTS 2084,0,2,2,1,0,1,0
ECON 2101,2,0,3,2,1,3,1
MSE 2001,2,3,0,1,1,2,0
ChBE 3200,1,2,1,0,0,1,1
FREN 1002,0,1,1,0,0,1,0
ISyE 3039,1,3,2,1,1,0,1
CELB 2200,0,1,0,1,0,1,0
```

5_whoHasConflicts.txt

```
List of class pairs and students with conflicts for that class pair:
For the course pair: HTS 2084 : ECON 2101
McDermitt, Tom, Junior
Del Vincent, Julie, Laure

For the course pair: HTS 2084 : MSE 2001
McDermitt, Tom, Junior
Del Vincent, Julie, Laure

For the course pair: HTS 2084 : ChBE 3200
Del Vincent, Julie, Laure

For the course pair: HTS 2084 : FREN 1002
```

```
For the course pair: HTS 2084 : ISyE 3039
McDermitt, Tom, Junior

For the course pair: HTS 2084 : CELB 2200

For the course pair: ECON 2101 : MSE 2001
McDermitt, Tom, Junior
Del Vincent, Julie, Laure
Jones, Matthew Charles

For the course pair: ECON 2101 : ChBE 3200
Del Vincent, Julie, Laure
Olsen, Kate

For the course pair: ECON 2101 : FREN 1002
Jones, Matthew Charles

For the course pair: ECON 2101 : ISyE 3039
McDermitt, Tom, Junior
Jones, Matthew Charles
Olsen, Kate
…
```

5_Schedule.txt

```
8:00 HTS 2084 FREN 1002 CELB 2200
9:00 ECON 2101
10:00 MSE 2001
11:00 ChBE 3200
12:00 ISyE 3039
```

5_ClassRolls.csv

```
HTS 2084:
"McDermitt, Tom, Junior"
"Del Vincent, Julie, Laure"

ECON 2101:
"McDermitt, Tom, Junior"
"Del Vincent, Julie, Laure"
"Jones, Matthew Charles"
"Olsen, Kate"

MSE 2001:
"McDermitt, Tom, Junior"
"Del Vincent, Julie, Laure"
"Jones, Matthew Charles"

ChBE 3200:
"Del Vincent, Julie, Laure"
"Olsen, Kate"

FREN 1002:
"Jones, Matthew Charles"
...
```

Console:

```
Please input Classes file name:
2017_6C.csv
Please input Students file name:
2017_6S.csv
```

2017_6C.csv

```
Course number,Course Title,Faculty
A1,A,Charles
B2,B,Bill
C3,C,Charles
D4,D,Doug
E5,E,Charles
F6,F,Flo
G7,G,Doug
H8,H,Charles
I9,I,Bill
```

2017_5S.csv

```
GTID,NAME,Course 1,Course 2,Course 3,Course 4
1,1,E5,I9,D4,C3
2,2,C3,B2,F6,I9
3,3,C3,B2,D4
4,4,B2,C3,H8,F6,E5
```

6_theConflictMatrix.csv

```
,A1,B2,C3,D4,E5,F6,G7,H8,I9
A1,0,0,0,0,0,0,0,0,0
B2,0,0,3,1,1,2,0,1,1
C3,0,3,0,2,2,2,0,1,2
D4,0,1,2,0,1,0,0,0,1
E5,0,1,2,1,0,1,0,1,1
F6,0,2,2,0,1,0,0,1,1
G7,0,0,0,0,0,0,0,0,0
H8,0,1,1,0,1,1,0,0,0
I9,0,1,2,1,1,1,0,0,0
```

6_whoHasConflicts.txt

```
List of class pairs and students with conflicts for that class pair:
For the course pair: A1 : B2

For the course pair: A1 : C3

For the course pair: A1 : D4

For the course pair: A1 : E5

For the course pair: A1 : F6

For the course pair: A1 : G7
```

```
For the course pair: A1 : H8

For the course pair: A1 : I9

For the course pair: B2 : C3
2
3
4

For the course pair: B2 : D4
3

For the course pair: B2 : E5
4

For the course pair: B2 : F6
2
4

For the course pair: B2 : G7

For the course pair: B2 : H8
4
…
```

6_Schedule.txt

```
8:00 A1 B2 G7
9:00 C3
10:00 D4 F6
11:00 E5
12:00 H8 I9
```

6_ClassRolls.csv

```
A1:

B2:
"2"
"3"
"4"

C3:
"1"
"2"
"3"
"4"

D4:
"1"
"3"

E5:
"1"
"4"

F6:
```

"2"
"4"

G7:

H8:
"4"

I9:
"1"
"2"

# *Appendix: Source Code*

```
/* Course.h
 * ECE2036 Final Project Part 2
 * 05-12-2017 Brighton ANCELIN
 *
 * Interface of Course class
 */

#ifndef COURSE_H_
#define COURSE_H_

#include<string>
#include<vector>

using namespace std;

class Course {
public:
    Course(string code, string title, string prof);
    Course(int invalidPlaceholder);
    virtual ~Course();
    static vector<Course> getAllCourses();
    static Course getCourseByCode(const string& code);
    static Course getCourseByEnumID(const int& enumID);
    string getCode() const;
    string getTitle() const;
    string getProf() const;
    int getEnumID() const;
    bool isValid() const;
private:
    static int enum_ctr;
    static vector<Course> all_courses;
    int enumID;
    string code;
    string title;
    string prof;
};

#endif /* COURSE_H_ */
```

```
/* Course.cpp
 * ECE2036 Final Project Part 2
 * 05-12-2017 Brighton ANCELIN
 *
 * Implementation of Course class
 */

#include<iostream>
#include<string>
#include "Course.h"

using namespace std;

/* Used for unique identifier */
int Course::enum_ctr{0};
/* Static vector of all unique course objects */
vector<Course> Course::all_courses{};

/* Instantiates a new course and gives it a unique integer identifier*/
Course::Course(string code, string title, string prof) :
        code{code}, title{title}, prof{prof}, enumID{enum_ctr} {
    // If this Course shares code with another (duplicate), do not add it to the
static all_courses vector
    for(const Course& c : getAllCourses()) {
        if(c.getCode() == code) {
            cout << "Duplicate course listing \"" << code << "\". Ignoring the
latter duplicate." << endl;
            return;
        }
    }
    enum_ctr++; // Increment unique identifier variable
    all_courses.push_back(*this); // Add this to all_courses vector
}

/* Constructor for an invalid course. Used as a flag, not an object, in practice */
Course::Course(int invalidPlaceholder) :
        code{"N/A"}, title{"N/A"}, prof{"N/A"}, enumID{-1} {}

Course::~Course() {}

vector<Course> Course::getAllCourses() {
    return all_courses;
}

Course Course::getCourseByCode(const string& code) {
    // Search for this course in vector of known courses
    for(const Course& c : all_courses) {
        if(code == c.code)
            return c;
    }
    // If course can't be found, say so and create an invalid one
    return Course{-1};
}

Course Course::getCourseByEnumID(const int& enumID) {
    return all_courses.at(enumID);
}

string Course::getCode() const {
```

```
    return this->code;
}
string Course::getTitle() const {
    return this->title;
}
string Course::getProf() const {
    return this->prof;
}
int Course::getEnumID() const {
    return this->enumID;
}

bool Course::isValid() const {
    return this->enumID != -1;
}
```

```
/* CourseConflictMap.h
 * ECE2036 Final Project Part 2
 * 09-11-2017 Brighton ANCELIN
 *
 * Interface of CourseConflictMap class
 */

#include<vector>
#include "StudentList.h"
#include "CoursePair.h"

#ifndef COURSECONFLICTMAP_H_
#define COURSECONFLICTMAP_H_

using namespace std;

class CourseConflictMap {
public:
    CourseConflictMap();
    virtual ~CourseConflictMap();
    void addConflict(const Course& c1, const Course& c2, Student student);
    StudentList at(const Course& c1, const Course& c2);
private:
    vector<vector<StudentList>> innerMapping;
};

#endif /* COURSECONFLICTMAP_H_ */
```

```
/* CourseConflictMap.cpp
 * ECE2036 Final Project Part 2
 * 09-11-2017 Brighton ANCELIN
 *
 * Implementation of CourseConflictMap class
 */

#include "Course.h"
#include "CourseConflictMap.h"

/* REQUIRES ALL COURSES TO HAVE ALREADY BEEN INITIALIZED
 * Initializes vector mapping to an upper triangular matrix, minus the diagonal
 */
CourseConflictMap::CourseConflictMap() {
    const size_t dim = Course::getAllCourses().size();
    for(size_t i{0}; i < dim-1; i++) {
        this->innerMapping.push_back(vector<StudentList>(dim-i-1, StudentList()));
    }
}

CourseConflictMap::~CourseConflictMap() {}

/* Adds a conflict to the inner mapping
 * Uses an upper triangular matrix, so the first course enum must be less than the
second
 */
void CourseConflictMap::addConflict(const Course& c1, const Course& c2, Student
student) {
    if(c1.getEnumID() < c2.getEnumID()) {
        this->innerMapping.at(c1.getEnumID()).at(c2.getEnumID()-c1.getEnumID()-
1).addStudent(student);
    } else {
        this->innerMapping.at(c2.getEnumID()).at(c1.getEnumID()-c2.getEnumID()-
1).addStudent(student);
    }
}

/* Gets a conflict from the inner mapping
 * Uses an upper triangular matrix, so the first course enum must be less than the
second
 */
StudentList CourseConflictMap::at(const Course& c1, const Course& c2) {
    if(c1.getEnumID() < c2.getEnumID()) {
        return this->innerMapping.at(c1.getEnumID()).at(c2.getEnumID()-
c1.getEnumID()-1);
    } else {
        return this->innerMapping.at(c2.getEnumID()).at(c1.getEnumID()-
c2.getEnumID()-1);
    }
}
```

```
/* CoursePair.h
 * ECE2036 Final Project Part 2
 * 09-11-2017 Brighton ANCELIN
 *
 * Interface of CoursePair class
 */

#ifndef COURSEPAIR_H_
#define COURSEPAIR_H_

#include "Course.h"

using namespace std;

class CoursePair {
public:
    CoursePair(Course c1, Course c2);
    virtual ~CoursePair();
    static vector<CoursePair> getAllUniqueCoursePairs();
    Course getCourse1() const;
    Course getCourse2() const;
private:
    Course c1;
    Course c2;
};

#endif /* COURSEPAIR_H_ */
```

```
/* CoursePair.cpp
 * ECE2036 Final Project Part 2
 * 09-11-2017 Brighton ANCELIN
 *
 * Implementation of CoursePair class
 */

#include "Course.h"
#include "CoursePair.h"

using namespace std;

CoursePair::CoursePair(Course c1, Course c2) :
        c1{c1}, c2{c2} {}

CoursePair::~CoursePair() {}

/* Creates upper triangular matrix entries, minus the diagonal, for a conflicts
matrix
 */
vector<CoursePair> CoursePair::getAllUniqueCoursePairs() {
    vector<CoursePair> uniqPairs;
    vector<Course> allCourses = Course::getAllCourses();
    // Iterate over all courses, except the last one
    for(size_t i{0}; i < allCourses.size()-1; i++) {
        // Start iterating one above i
        for(size_t j{i+1}; j < allCourses.size(); j++) {
            uniqPairs.push_back(CoursePair(allCourses[i], allCourses[j]));
        }
    }
    return uniqPairs;
}

Course CoursePair::getCourse1() const {
    return this->c1;
}
Course CoursePair::getCourse2() const {
    return this->c2;
}
```

```
/* Main.cpp
 * ECE2036 Final Project Part 2
 * 05-12-2017 Brighton ANCELIN
 *
 * Primary driving program
 */

#include<iostream>
#include<fstream>
#include<vector>
#include<string>
#include<sstream>
#include "Course.h"
#include "CourseConflictMap.h"
#include "CoursePair.h"
#include "Student.h"
#include "StudentList.h"

using namespace std;

vector<string> tokenizeFile(istream& stream);

int main() {
    cout << "Please input Classes file name: \n";
    string classesFile;
    getline(cin, classesFile);
    cout << "Please input Students file name: \n";
    string studentsFile;
    getline(cin, studentsFile);

    string dummyStr;

    ifstream courseInStream{classesFile, ios::in};
    if(!courseInStream) {
        cerr << "File \"" << classesFile << "\" could not be opened." << endl;
        exit(EXIT_FAILURE);
    }

    getline(courseInStream, dummyStr);
    string str;
    // Build all Courses (full vector retained as a static member of the Course
class)
    while(getline(courseInStream, str)) {
        stringstream strStream{str};
        vector<string> tokens = tokenizeFile(strStream);
        if(tokens.size() < 3) {
            cout << "File \"" << classesFile << "\" has an incomplete line. This
line will be ignored." << endl;
            continue;
        }
        string code = tokens.at(0);
        string title = tokens.at(1);
        string prof = tokens.at(2);
        Course{code, title, prof};
    }

    ifstream studentInStream{studentsFile, ios::in};
    if(!studentInStream) {
        cerr << "File \"" << studentsFile << "\" could not be opened." << endl;
```

```
        exit(EXIT_FAILURE);
    }

    vector<Student> students;
    getline(studentInStream, dummyStr);
    // Build all students and add them to local vector
    // Also build rosters (vector of StudentList) for each course
    vector<StudentList> rosters =
vector<StudentList>(Course::getAllCourses().size(), StudentList());
    while(getline(studentInStream, str)) {
        stringstream strStream{str};
        vector<string> tokens = tokenizeFile(strStream);
        if(tokens.size() < 3) {
            cout << "File \"" << studentsFile << "\" has an incomplete line. This
line will be ignored." << endl;
            continue;
        }
        string gtid = tokens.at(0);
        string name = tokens.at(1);
        vector<Course> curStudCourses;
        for(size_t i{2}; i < tokens.size(); i++) {
            Course c{Course::getCourseByCode(tokens.at(i))};
            if(c.isValid()) {
                curStudCourses.push_back(Course::getCourseByCode(tokens.at(i)));
            } else {
                cout << "Student \"" << name << "\" requested course \"" <<
tokens.at(i) << "\", which is not offered. Ignoring request." << endl;
            }
        }
        Student curStudent{gtid, name, curStudCourses};
        students.push_back(curStudent);
        for(const Course& c : curStudCourses) {
            // Safeguard against extra-courses being added by students and indexing
beyond the vector size
            while (c.getEnumID() >= rosters.size()) {
                rosters.push_back(StudentList());
            }
            rosters.at(c.getEnumID()).addStudent(curStudent);
        }
    }

    // ONLY CREATE THIS WHEN COURSES ARE FULLY INITIALIZED
    CourseConflictMap conflicts;
    // For all students, add all their conflicts to the conflicts object
    for(const Student& student : students) {
        for(const CoursePair& cpair : student.getCourseConflicts()) {
            conflicts.addConflict(cpair.getCourse1(), cpair.getCourse2(), student);
        }
    }

    // Add edges to the conflict matrix for professor conflicts
    CourseConflictMap conflictsProfs{conflicts};
    for(size_t i{0}; i < Course::getAllCourses().size(); i++) {
        for(size_t j{i+1}; j < Course::getAllCourses().size(); j++) {
            Course c1{Course::getCourseByEnumID(i)};
            Course c2{Course::getCourseByEnumID(j)};
            if(c1.getProf() == c2.getProf()) {
                conflictsProfs.addConflict(c1, c2, Student{"N/A", c1.getProf(),
vector<Course>()});
```

```
            }
        }
    }

    // Build all timeslots of courses
    // In effect, each course goes through all currently existing timeslots and
checks to see if the course can go in each one.
    // Once the course finds a timeslot the course can join, the course joins the
timeslot.
    // If the course never finds such a timeslot, the course will be placed in a
new timeslot
    vector<vector<Course>> timeslots;
    for(Course c : Course::getAllCourses()) {
        bool hasBeenInserted = false;
        for(vector<Course>& curBlock : timeslots) {
            // If the timeslot is full, skip it
            if(curBlock.size() >= 6) {
                continue;
            }
            // If the course has already found a timeslot, stop searching for a
timeslot
            if(hasBeenInserted) {
                break;
            }
            // If any course currently in the timeslot conflicts with this new
course, break and try the next timeslot
            bool canInsertCurCourse = true;
            for(Course c2 : curBlock) {
                if(conflictsProfs.at(c, c2).getSize() != 0) {
                    canInsertCurCourse = false;
                    break;
                }
            }
            // If no course currently in the timeslot conflicts with this new
course, add the new course to the timeslot
            if(canInsertCurCourse) {
                curBlock.push_back(c);
                hasBeenInserted = true;
            }
        }
        // If no currently existing time slot could hold this new course, give it
its own timeslot
        if(!hasBeenInserted) {
            vector<Course> newBlock(1, c);
            timeslots.push_back(newBlock);
        }
    }

    ofstream conflMatFile{"theConflictMatrix.csv", ios::out};
    if(!conflMatFile) {
        cerr << "File theConflictMatrix.csv couldn't be made." << endl;
        exit(EXIT_FAILURE);
    }

    // Write the header to the file
    for(const Course& c1 : Course::getAllCourses()) {
        conflMatFile << "," << c1.getCode();
    }
    conflMatFile << endl;
```

```
    for(const Course& c1 : Course::getAllCourses()) {
        // Write the side column entries
        conflMatFile << c1.getCode();
        for(const Course& c2 : Course::getAllCourses()) {
            if(c1.getEnumID() == c2.getEnumID()) {
                // Leave the cell 0 for impossible conflicts
                conflMatFile << ",0";
                continue;
            }
            // Output the conflict count
            conflMatFile << "," << conflicts.at(c1, c2).getSize();
        }
        conflMatFile << endl;
    }

    ofstream whoHasConflFile{"whoHasConflicts.txt", ios::out};
    if(!whoHasConflFile) {
        cerr << "File whoHasConflicts.txt couldn't be made." << endl;
        exit(EXIT_FAILURE);
    }

    // Print header to file
    whoHasConflFile << "List of class pairs and students with conflicts for that
class pair:\n";
    for(const CoursePair& cpair : CoursePair::getAllUniqueCoursePairs()) {
        // Print current course pair title
        whoHasConflFile << "For the course pair: " << cpair.getCourse1().getCode()
                << " : " << cpair.getCourse2().getCode() << endl;
        // Print each student's name below the title
        for(const Student& student : conflicts.at(cpair.getCourse1(),
cpair.getCourse2()).getInnerVec()) {
            whoHasConflFile << student.getName() << endl;
        }
        // Separate regions with an extra newline character
        whoHasConflFile << endl;
    }

    ofstream scheduleFile{"Schedule.txt", ios::out};
    if(!scheduleFile) {
        cerr << "File Schedule.txt couldn't be made." << endl;
        exit(EXIT_FAILURE);
    }

    // Print timeslot concerns to the console
    if(timeslots.size() > 24) {
        cout << "There's not enough time in a day for this many classes" << endl;
        cout << "It is necessary to remove to some classes, conflicts, or find more
classrooms" << endl;
    } else if(timeslots.size() > 16) {
        cout << "Classes will run past midnight. That might upset some students" <<
endl;
    } else if(timeslots.size() > 14) {
        cout << "Classes will run past 10PM. That might upset some students." <<
endl;
    }

    size_t blockHour{8};
    for(const vector<Course>& curBlock : timeslots) {
        // Print the block hour
```

```
        scheduleFile << (blockHour++ % 24) << ":00";
        // Print all courses at this block hour
        for(const Course& c : curBlock) {
            scheduleFile << " " << c.getCode();
        }
        scheduleFile << endl;
    }

    ofstream classRollsFile{"ClassRolls.csv", ios::out};
    if(!classRollsFile) {
        cerr << "File ClassRolls.csv couldn't be made." << endl;
        exit(EXIT_FAILURE);
    }

    for(size_t i{0}; i < rosters.size(); i++) {
        // Print course name
        classRollsFile << Course::getCourseByEnumID(i).getCode() << ":" << endl;
        // Print all students in the course, enclosed in quotes to keep a single
column
        for(const Student& curStud : rosters.at(i).getInnerVec()) {
            classRollsFile << "\"" << curStud.getName() << "\"" << endl;
        }
        classRollsFile << endl;
    }
}

/* Tokenizes a CSV file line, where columns are separated by commas, rows by
newlines, and quotes encapsulate text directly*/
vector<string> tokenizeFile(istream& stream) {
    vector<string> tokens;
    string str;
    // Read up to next instance of opening quote
    while(getline(stream, str, '"')) {
        stringstream strStream{str};
        // Read from strStream up to first comma
        while(getline(strStream, str, ',')) {
            // Push token
            tokens.push_back(str);
        }
        // If a closing quote is found (i.e. not EOF)
        if(getline(stream, str, '"')) {
            // Push token
            tokens.push_back(str);
            // Remove left-over following delimiter
            if(',' == stream.peek()) {
                getline(stream, str, ',');
            }
        }
    }
    return tokens;
}
```

```
/* Student.h
 * ECE2036 Final Project Part 2
 * 09-11-2017 Brighton ANCELIN
 *
 * Interface of Student class
 */

#ifndef STUDENT_H_
#define STUDENT_H_

#include<string>
#include<vector>
#include "Course.h"
#include "CoursePair.h"

using namespace std;

class Student {
public:
    Student(string gtid, string name, vector<Course> courseRequests);
    virtual ~Student();
    string getGTID() const;
    string getName() const;
    vector<Course> getCourseRequests() const;
    vector<CoursePair> getCourseConflicts() const;
private:
    string gtid;
    string name;
    vector<Course> courseRequests;
};

#endif /* STUDENT_H_ */
```

```
/* Student.cpp
 * ECE2036 Final Project Part 2
 * 09-11-2017 Brighton ANCELIN
 *
 * Implementation of Student class
 */

#include "Course.h"
#include "CoursePair.h"
#include "Student.h"

Student::Student(string gtid, string name, vector<Course> courseRequests) :
        gtid{gtid}, name{name}, courseRequests{courseRequests} {}

Student::~Student() {}

string Student::getGTID() const {
    return this->gtid;
}

string Student::getName() const {
    return this->name;
}

vector<Course> Student::getCourseRequests() const {
    return this->courseRequests;
}

/* Returns all conflicts this student might have */
vector<CoursePair> Student::getCourseConflicts() const {
    vector<CoursePair> conflicts;
    // Iterate over all courses, except the last one
    for(size_t i{0}; i < this->courseRequests.size()-1; i++) {
        // Start iterating one above i
        for(size_t j{i+1}; j < this->courseRequests.size(); j++) {
            conflicts.push_back(CoursePair(static_cast<Course>(this-
>courseRequests[i]), static_cast<Course>(this->courseRequests[j])));
        }
    }
    return conflicts;
}
```

```
/* StudentList.h
 * ECE2036 Final Project Part 2
 * 09-11-2017 Brighton ANCELIN
 *
 * Interface of StudentList class
 */

#include<vector>
#include "Student.h"

#ifndef STUDENTLIST_H_
#define STUDENTLIST_H_

using namespace std;

class StudentList {
public:
    StudentList();
    virtual ~StudentList();
    Student operator[](const int& index) const;
    Student& operator[](const int& index);
    void addStudent(Student student);
    vector<Student> getInnerVec() const;
    int getSize() const;
private:
    vector<Student> innerVec;
};

#endif /* STUDENTLIST_H_ */
```

```
/* StudentList.cpp
 * ECE2036 Final Project Part 2
 * 09-11-2017 Brighton ANCELIN
 *
 * Implementation of StudentList class
 */

#include "StudentList.h"

StudentList::StudentList() {}

StudentList::~StudentList() {}

Student StudentList::operator[](const int& index) const {
    return this->innerVec[index];
}

Student& StudentList::operator[](const int& index) {
    return this->innerVec[index];
}

void StudentList::addStudent(Student student) {
    this->innerVec.push_back(student);
}

vector<Student> StudentList::getInnerVec() const {
    return this->innerVec;
}

int StudentList::getSize() const {
    return this->innerVec.size();
}
```