CMU Final Exam Scheduling

Operation Research II Project

By Zeheng Xu, Yige Zhang, Zhehe Qiao, Samuel Jia

Table of Content

Abstract

This report investigates the optimal final exam schedule for Carnegie Mellon University (CMU) Fall 2017. We seek to minimize the total number of conflicts CMU undergraduate students faced. The conflicts are defined by having two exams at the same time slot and having three exams in 24 hours. This problem will be beneficial to the whole community if appropriately solved. Given the large number of inputs and uncertainties, we categorized it as an NP-hard problem and have made several assumptions for building integer problem. After applying the mathematical reasoning we've learned in Operational Research class, we use programming to help implement our theory to final results.

Introduction

Exam scheduling is a challenging task that universities and colleges face several times every year. There is a moderate possibility that students will deal with final exam conflicts at some points in their college life. In this project, we aim to explore the final exam scheduling at CMU for 2017 Fall by generating a possible optimal exam schedule that minimizes the conflicts. Given a large amount of information(students, classroom availability, etc.) in a short period, exam scheduling problem can be an NP-hard problem. The integer program we built is based on several assumptions on inputs to construct the linear integer programming. We constructed reasonable constraints to this problem that allow us to solve this problem with mathematical tools we learned in class.

Without loss of generality and be as realistic as possible, the model is built on information of Carnegie Mellon University. We generate data on exams each student should take, size of classrooms, exams and corresponding classrooms. With the adoption of python programming, we are able to implement different methods of scheduling final exams. After trying various algorithms and methods of implementation, we are able to reduce the percentage of conflicts from 11.16% to 6.67%.

We also recognized some potential field that could be further improved if given more time. First of all, we should spread out students' exams to provide students with more review time for each exam. Secondly, if we take into consideration the fact that Students within each department tend to take similar courses, conflicts can be further reduced. Finally, the capacity of each classroom could be more precise comparing to having three big categories we had for integer programming.

Assumptions

First of all, based on reality, we assume that all exams should be completed in a week, with Wednesday being reading day and no exams. This leaves us with six days for scheduling final exams. Secondly, each day consists of 3 sessions: morning, afternoon and evening, which leads us to total 18 time slots. In addition, to build our first implementation, we assume all the students want to finish their exams as early as possible. We then make adjustments based on the assumption.

1. Time

First of all, based on reality, we assume that all exams should be completed in a week, with Wednesday being reading day and now exams. This leaves us with 6 days for scheduling final exams. Secondly, each day consists of 3 sessions: morning, afternoon and evening, which leads us to total 18 time slots. In addition, to build our first implementation, we assume all the students want to finish their exams as early as possible. We then make adjustment based on the assumption.

2. Students

From basic statistics of CMU, we set the number of undergraduate students to be 6100 students. We assume that no one will overload and thus each student will take less than 5 exams. Furthermore, we are not dividing students into different departments. Since we are unable to retrieve the real data on what classes each student is taking, we randomly assign no more than 5 classes for each student to take.

3. Exams

From CMU 2017 Fall final exam information we found online, we set the number of exams to be 465 exams for the semester. Since the exams are randomly taken by students, we expect the number of conflicts will arise from this assumption. In addition, we use real data of max enrollment on those 465 exams as our bar for maximum number of students that can be enrolled for each exam.

4. Classrooms

From public published data and resources we've found on past final exam classrooms, we decide there will be 93 classrooms available during the final exam period. Since each exam has difference enrollment, we also divide the classrooms into three categories based on capacity:

- < 50 people: 70
- 50 - 100 people: 10
- > 100 people: 13

Exams taken by a large amount of students (e.g. more than 100 students) will only be taken in large classrooms. So one exam will not be broken into several smaller classrooms.

Formulation

We start our process of formulating a linear integer programming by determining the objective function and then explore the constraints. As discussed above, our main goal is to minimize the total number of conflicts that students face during final week. This part falls into two conflicts: students who are taking exams at the same time slots and students who are taking three exams within 24 hours. For constraints, we have developed 5 basic constraints to guide our integer program. Firstly, at the same time slot, the total number of exams should not exceed total number of classrooms. Secondly, at the same time slot, in classroom k, there's at most one exam. Thirdly, capacity of classrooms for an exam should be able to accommodate all students taking the exam. Fourthly, each student can take less than 5 exams. Lastly, for each exam, the number of students is less than the maximum course capacity.

Objective Function

1. Variables

k: classroom $0 < k \leq 93$

i: student $0 < i \leq 6100$

j: course number $0 < j \leq 465$

t: time slots $0 < t \leq 18$

E_j: capacity of course j

C_k: max capacity of classroom k

$\mathbf{A_{i,j}}$ = 1, if student i takes exam j

0, otherwise

$\mathbf{P_{j,k,t}}$ = 1, if exam j takes place in classroom k at time t

0, otherwise

$\mathbf{X_{i,j,k,t}}$ = 1, if student i takes exam j in classroom k at time t

0, otherwise

We aim to define the most important variables: index of classrooms, index of students, index of course number and index of time slots. We also listed the capacity of course j and max capacity of classroom k for the ease of future expression.

In addition, we define 3 conditional variables. $\mathbf{A_{i,j}}$ stands for the possibility that student i takes exam j. $\mathbf{P_{j,k,t}}$ stands for the possibility that exam j takes place in classroom k at time

t. $\mathbf{X_{i,j,k,t}}$ stands for the possibility of student i takes exam j in classroom k at time t. With all the variables we have defined, we will move into constructing objective function.

2. Objective function -- Conflict #1

$$Y_{i,t} = \sum_{j=1}^{465} \sum_{k=1}^{93} X_{i,j,k,t}$$

$$Z_{i,t} \geq 0$$

$$Z_{i,t} \geq Y_{i,t} - 1$$

As mentioned previously, the first conflict is that one student takes multiple exams at the same time slot, which is the most common conflict. Xi,j,k,t undertakes students, exams, classrooms and times, while $0 < i \leq 6100$, $0 < j \leq 465$, $0 < k \leq 93$, $0 < t \leq 18$ being the individual bounds of variables. We sum Xi,j,k,t over j (course) and k (classroom) to get the number of exams each student has for at each time slot, noted by Yi,t. We further define a maximum relationship between Yi,t - 1 and 0 by adopting another variable, namely Zi,t. Zi,t takes the larger of 0 and (Yi,t - 1). Given the conflict of each student at each time slot, we them sum up Zi,t over i (students) and t (times slots) to get the total number of conflicts of students having multiple exams at the same time slot.

Conflict #1: $$\sum_{i=1}^{6100} \sum_{t=1}^{18} Z_{i,t}$$

3. Objective function -- Conflict #2

The second conflict is defined as students taking 3 exams within 24 hours time frame. This conflict is more complex than the conflict since we need to trace back to three consecutive exams for implementation, comparing to looking at single exam for conflict #1. To start off, we introduced another variable Ui,t which takes the following values:

$\mathbf{U_{i,t}}$ = 1, if student i has 3 exams in 24 hours

0, otherwise

Then we define a new variable related to $\mathbf{U_{i,t}}$

$$W_{i,t} = \sum_{j=1}^{465} \sum_{k=1}^{93} X_{i,j,k,t} + X_{i,j,k,t+1} + X_{i,j,k,t+2}$$

$$U_{i,t} \geq 0$$

$$U_{i,t} \geq W_{i,t} - 2$$

Similar to $\mathbf{Y_{i,t}}$, $\mathbf{W_{i,t}}$ is the number of exams one students would take in three consecutive time slots. $\mathbf{U_{i,t}}$ is introduced to compute the maximum number of $\mathbf{W_{i,t}}$ - 2 and 0, while $\mathbf{W_{i,t}}$ - 2 stands for the number of conflicts and 0 stands for no conflict.

Next step, we introduced another variable V_i , which measures the number of 3 exams taken by student i in 24 hours.

$$V_i = \sum_{t=1}^{16} U_{i,t}$$

$$R_i \leq 1$$

$$R_i \leq V_i$$

R_i, on the contrary, is introduced to compute the minimum between 1 and V_i. If R_i takes the value 1, it means student i does have conflict #2.

Finally, we sum R_i over i to get the total number of conflicts of students having multiple exams within 24 hours.

Conflict #2: $\sum_{i=1}^{6100} R_i$

In all, we have the objective function:

$$\text{Min} \sum_{i=1}^{6100} \sum_{t=1}^{18} Z_{i,t} + \sum_{i=1}^{6100} R_i$$

Constraints

Constraint #1

$$\sum_{j=1}^{465}\sum_{k=1}^{93} P_{j,k,t} \leq 93$$

We define the first constraint to be at time t, the total number of exams should not exceed total number of classrooms. Pj,k,t , as defined before, takes the value 1 if exam j takes place in classroom k at time t. We sum Pj,k,t over j and k to get the total number of exams happened at time t. The sum should never exceed the maximum number of classrooms, 93.

Constraint #2

$$\sum_{j=1}^{465} P_{j,k,t} \leq 1$$

We define the second constraint to be at time t, in classroom k, there's at most one exam. In this way, for all k, t, we sum over j (taking the value 1 to 465) will get the number of exams happened at time t and classroom 1. The sum should never exceed 1, which means we could only have a classroom with one exam or no exam.

Constraint #3

$$\sum_{i=1}^{6100}\sum_{j=1}^{465} X_{i,j,k,t} \leq C_k, \quad \text{for all } k, t$$

The third constraint is defined as the capacity of classrooms for an exam should be able to accommodate all students taking the course. We take the sum of Xi,j,k,t over i and j to get the total number of students taking one exam in classroom k at time t. The sum should never exceed Ck, the capacity of each classroom.

Constraint #4

$$\sum_{j=1}^{465} A_{i,j} \leq E_j$$

Our last constraint is based on capacity of each courses. We need to make sure that for each class, the number of students is less than the maximum course capacity. We sum Ai, j over i to get the total number of students taking each courses. The sum should be less or equal to Ej, namely the capacity of each course j.

Now, we have constructed our complete linear integer problem. Next step is to implement our algorithms and hopefully obtain the best solution.

Algorithms

Initial schedule 1:

In this schedule, we assume that largest-size courses will most likely cause conflicts. The idea is that we first assign time slots to largest classes to avoid conflicts, then fit smaller classes in the rest of time slots and classrooms. The specific steps are as following:
Fit all courses with over 100 students in classrooms that can accommodate over 100 students first. Then we fit courses of size 50-100 students, then courses with less than 50 students.
It turns out that we only used 17 time-slots. The total number of conflicts is 3349.

Improvement 1:
Since we only used first 17 time-slots, we can fit classes with most conflicts to the last slot without causing further conflicts. On average, each day should have 26 classes. So we choose the 26 classes that create the most conflicts and then put them to time slot 18. The result is 2933 conflicts in total.

Improvement 2:
The problem with this algorithm is that if we choose two classes in the same time slot, their conflicts will remain if we put those two classes together to time slot 18. On the second try, for each time slot, we pick up the class with most conflicts and put them to the 18th time slot. And this leads us to a more optimal result. Consequently, we end up with 2729 conflicts in total.

Initial Schedule 2:

For the trials on initial schedule 1, we found that for schedules with classes arranged more evenly, there are fewer conflicts. Then our 2nd initial schedule is to assign classes to time slots randomly and uniformly. To our surprise, we only have 2504 conflicts for this schedule.

Improvement 1(Swap):

Since classes should always be put evenly over the 18 time slots, we choose to swap classes that causes the most schedules. The algorithm is to swap classes that cause the most conflicts and the second most conflicts. After getting an improved schedule, we apply this algorithm again till cannot proceed. The final result is 2349, which is a much better result than the previous trials.

Improvement 2:

We checked the deadlock causing this issue. The problem is that in the last result schedule that causes the deadlock, after swapping classes, we will have a resulting schedule with the same two classes that are causing the most and the second most conflicts. Then we are trapped in an infinite loop. To get out of this infinite loop, we choose the class that causes the most number of conflicts to swap with the max-conflict class. With this algorithm, we get 2051 conflicts and go to some infinite loop again.

Improvement 3:

The problem is almost the same. Although we are guaranteed that we would not have two same swap courses over and over again, we would face a problem of having the same three schedules going to infinite loop. Our solution is to keep a checker out of the loop, and every time we found infinite loop on three schedules, we swap the second and the third classes that are causing the

most conflicts. This algorithm is very robust since we have three checkers. If we fail the first one, we will swap the course that leads to most conflicts and the course that leads to the third most conflicts. The checker will fail only if we continuously get the same schedule for three possible solutions. Note that each time we get out of the loop, the checkers are reset. We minimize the total number of conflicts to 1789, then after running our program for a while, the number of conflicts jumps up back to over 2000.

Improvement 4:

The problem with the previous improvement is simply that around 1800, the number of conflicts gets pretty close to optimal so that switching courses with the most conflicts are very likely to cause the number of conflicts increase. We add another checker in the loop to choose the schedule only if the schedule would increase less than 15 conflicts (define reducing conflicts as negative). The final result is that we have minimized the number of conflicts to 1668. The final result is decent since the problem involves over 30000 person-times. And we successfully reduced the number of conflicts from 3349 to 1668, which is over 50 percent.

Implementation

Given the large amount of variables and inputs involved in our linear programming problem, we could not fully execute it as a math problem. We seek the help of python for sample scheduling. For the sake of clearness, we will use the same convention for variables to represent number of students, number of classrooms, etc., as discussed in Objective Function section.

1. Data Structure

We represent each student as a list of Boolean values. Then, we store all the lists that represent students in a higher level list. For each list S[i], S[i] = [id, C, b1, b2, b3…, b465]. Id is student's index in the table. C = total number of classes that the student is taking. For b1, b2...b465, each bi = True if student is taking class i, and false otherwise. The total length of 2D list is 6100, as the number of students.

For schedules, we have another 2D list G. Class i's information is stored in G[i], respectively. G[i] = [class ID, number of students, time slot, i]. Time slot is a number from 0 to 18. Class ID is the class number, such as 21393 for OR2.

2. Data Generation

Since students' information is not accessible, we used functions to randomly assign classes to each student according to available information about courses. The algorithm is to go through each class and in list G. In each iteration i, randomly choose b students

with classes less than 5, where b is the total number of students taking class G[i]. Then mark S[d][i] with false. Notice S[d] is student's b class information. As a result, we have a quite randomly distributed data set for students' class information. As for late use in computation, the student's information is stored in another list S', where S'[d] represent dth student. The first 2 entries are kept the same. For class taken, instead of Booleans, we replace them with class indexes in C. For example, if student 4 is taking 5th class, 10th class, 13th class, 56th class, his list in S' will be [4, 4, 5, 10, 13, 56].

3. Find conflicts and Change Schedule

This data structure makes it relatively simple to find the total number of conflicts for given schedule. To compute total conflicts, we simply go through the list S' for each student. On ith iteration, we will access class by class information stored in S'[i]. Referring back to example of student 4: the time slots of his classes are:

H = G[S'[4][3]][2], G[S'[4][4]][2], G[S'[4][5]][2], G[S'[4][6]][2]

Note that G[S'[4][3]] is 5, which is class index in G. and G[5][2] is its time slot. Then sort the list H by time slot. We increment total number of conflicts by 1 if there is a equal time slot in class j, class j+1 in sorted student's classes, since conflicts can only happen between continuous classes after sorting. As for the use of algorithms, it is usually need to keep track of how many conflicts each class is causing. Then we just have a separate list of length 465 and adding conflicts respectively to each index of each class. In order to change schedule, we will just change time slot numbering on schedule list G. If the size need to be changed, we assign new classroom as well.

Conclusions

Our project successfully reduced over one-half of the total conflicts in the initial schedule and our algorithm still shows the potential to do better. Attached in appendix E is the optimal schedule we were able to obtain. The greatest challenge we confronted during the project was due to the large amount of data size. As a result, we were not able to apply linear programming directly to this problem. We should also bear in mind that our optimization is based on some simplification and assumption. In practice, the problem is more complicated with additional possibility and combinations to consider. In order to obtain a more practical and insightful solution, here are some of the improvements we shall consider: data-wise, students' class schedule would be more complicated in real-life scenario, and classroom capacity shall be updated with accurate numbers. As for algorithm, we hit a bottleneck that for every fifty rounds of computations we end up in an infinite loop that gave us repeated solutions. Thus, our result would be further improved if we could find a solution to breaking infinite loop of larger size.

Reference

-Kweon Woo Jung, Min Jung Kim, Jisu Kwak, KangHa Lim, CMU Fall 2010 Final Exam Schedule, https://www.math.cmu.edu/~af1p/Teaching/OR2/Projects/P14/21393_G7.pdf

-Batenburg K. and Palenstijn W., “A New Exam Timetabling Algorithm,” Leiden Institute of Advanced Computer Science (LIACS), http://visielab.ua.ac.be/staff/batenburg/papers/ba pa_bnaic_2003.pdf.

-Mohammad Malkawi1 , Mohammad Al-Haj Hassan , and Osama Al-Haj Hassan,A New Exam Scheduling Algorithm Using Graph Coloring

-Professor Alan Frieze

-CMU Registration Office

Appendix

A. Initial Class Schedule

Course	Size	Slot	Room#
48250	020	01	01
48315	020	01	02
48324	020	01	03
48357	020	01	04
48524	020	01	05
48558	020	01	06
48568	020	01	07
48635	020	01	08
48729	020	01	09
48734	020	01	10
48740	020	01	11
48771	020	01	12
60157	020	01	13
60220	020	01	14
60424	020	01	15
03121	020	01	16
03124	020	01	17
03125	020	01	18
03133	020	01	19
03151	020	01	20
03201	020	01	21
30220	020	01	22
30301	020	01	23
03320	020	01	24
03327	020	01	25
03362	020	01	26
03401	020	01	27
03439	020	01	28
03511	020	01	29
03534	020	01	30
03709	020	01	31
03711	020	01	32
03727	020	01	33
42101	020	01	34
42202	020	01	35
42302	020	01	36
42341	020	01	37
42401	020	01	38
42620	020	01	39
42671	020	01	40
42674	020	01	41
42773	020	01	42
70122	020	01	43
70208	020	01	44
70311	020	01	45
70318	020	01	46
70341	020	01	47
70365	020	01	48
70391	020	01	49
70398	020	01	50
70451	020	01	51
70460	020	01	52
70477	020	01	53

70497	020	01	54
52400	020	01	55
52401	020	01	56
62110	020	01	57
62150	020	01	58
62633	020	01	59
06100	020	01	60
06221	020	01	61
06321	020	01	62
06323	020	01	63
06423	020	01	64
06614	020	01	65
06623	020	01	66
06703	020	01	67
06705	020	01	68
06815	020	01	69
09105	020	01	70
09106	020	02	01
09107	020	02	02
09207	020	02	03
09214	020	02	04
09217	020	02	05
09219	020	02	06
09221	020	02	07
09231	020	02	08
09344	020	02	09
09347	020	02	10
09536	020	02	11
09721	020	02	12
09736	020	02	13
39402	020	02	14
39613	020	02	15
12100	020	02	16
12335	020	02	17
12355	020	02	18
12411	020	02	19
12620	020	02	20
12629	020	02	21
12635	020	02	22
12651	020	02	23
12702	050	01	71
12704	050	01	72
12709	050	01	73
12712	050	01	74
12720	050	01	75
12729	050	01	76
12741	050	01	77
12751	050	01	78
12752	050	01	79
02261	050	01	80
02601	050	02	71
02613	050	02	72
02711	050	02	73
15110	050	02	74
15112	050	02	75
15121	050	02	76
15122	050	02	77
15150	050	02	78
15151	050	02	79
15210	050	02	80
15214	050	03	71
15251	050	03	72
15313	050	03	73
15317	050	03	74
15322	050	03	75
15351	050	03	76
15381	050	03	77
15410	050	03	78
15414	050	03	79
15421	050	03	80

15437	050	04	71
15440	050	04	72
15441	050	04	73
15445	050	04	74
15451	050	04	75
15456	050	04	76
15458	050	04	77
15462	050	04	78
15463	050	04	79
15487	050	04	80
15605	050	05	71
15614	050	05	72
15622	050	05	73
15637	050	05	74
15640	050	05	75
15641	050	05	76
15645	050	05	77
15650	050	05	78
15651	050	05	79
15657	050	05	80
15662	050	06	71
15663	050	06	72
15681	050	06	73
15852	050	06	74
15858	050	06	75
15862	050	06	76
15883	050	06	77
66161	050	06	78
66221	050	06	79
67240	050	06	80
67262	050	07	71
67328	050	07	72
67329	050	07	73
73102	050	07	74
73230	050	07	75
73240	050	07	76
73328	050	07	77
73347	050	07	78
73366	050	07	79
73374	050	07	80
73421	050	08	71
18090	050	08	72
18202	050	08	73
18220	050	08	74
18240	050	08	75
18290	050	08	76
18300	050	08	77
18340	050	08	78
18349	050	08	79
18370	050	08	80
18372	050	09	71
18422	050	09	72
18492	050	09	73
18618	050	09	74
18622	050	09	75
18631	050	09	76
18639	050	09	77
18643	050	09	78
18644	050	09	79
18648	050	09	80
18650	050	10	71
18715	050	10	72
18730	050	10	73
18749	050	10	74
18751	050	10	75
18755	050	10	76
18765	050	10	77
18771	050	10	78
18781	050	10	79
18785	050	10	80

18792	050	11	71
18793	050	11	72
18794	050	11	73
18847	050	11	74
19101	050	11	75
19301	050	11	76
19403	050	11	77
19424	050	11	78
19440	050	11	79
19462	050	11	80
19472	050	11	89
19638	050	11	90
19639	050	11	91
19691	050	11	92
19713	050	11	93
19717	050	12	71
19740	050	12	72
19751	050	12	73
76205	050	12	74
76223	050	12	75
76239	050	12	76
76315	050	12	77
76337	050	12	78
76366	050	12	79
76375	050	12	80
76389	050	12	81
76396	050	12	82
76441	050	12	83
76450	050	12	84
76457	050	12	85
76481	050	12	86
76766	050	12	87
76775	050	12	88
76789	050	12	89
76791	050	12	90
76796	050	12	91
76841	050	12	92
76850	050	12	93
76857	050	13	71
76881	050	13	72
53353	050	13	73
53451	050	13	74
53740	050	13	75
53751	050	13	76
79212	050	13	77
79229	050	13	78
79236	050	13	79
79259	050	13	80
79261	050	13	81
79265	050	13	82
79277	050	13	83
79299	050	13	84
79310	050	13	85
79327	050	13	86
79352	050	13	87
05341	050	13	88
05391	050	13	89
05410	050	13	90
05430	050	13	91
05431	050	13	92
05499	050	13	93
05610	050	14	71
05630	050	14	72
05631	050	14	73
05814	050	14	74
05823	050	14	75
05891	050	14	76
05899	050	14	77
14642	050	14	78
14741	050	14	79

14848	050	14	80
84326	050	14	81
84369	050	14	82
84669	050	14	83
08631	050	14	84
08672	050	14	85
08722	050	14	86
08736	050	14	87
11291	050	14	88
11411	050	14	89
11492	050	14	90
11611	050	14	91
11676	050	14	92
11692	050	14	93
11711	050	15	71
11751	050	15	72
11777	050	15	73
11785	050	15	74
11792	050	15	75
11927	050	15	76
10601	050	15	77
10701	050	15	78
27100	050	15	79
27201	050	15	80
27215	050	15	81
27301	050	15	82
27324	050	15	83
27432	050	15	84
27502	050	15	85
27766	050	15	86
27797	050	15	87
27799	050	15	88
21111	050	15	89
21112	050	15	90
21120	050	15	91
21122	050	15	92
21127	050	15	93
21128	050	16	71
21228	050	16	72
21235	050	16	73
21240	050	16	74
21241	050	16	75
21242	050	16	76
21256	050	16	77
21257	050	16	78
21259	050	16	79
21260	050	16	80
21268	050	16	81
21300	050	16	82
21301	050	16	83
21325	050	16	84
21341	050	16	85
21355	050	16	86
21356	050	16	87
21369	050	16	88
21370	050	16	89
21371	050	16	90
21373	050	16	91
21378	050	16	92
21441	050	16	93
21602	050	17	71
21603	050	17	72
21632	100	01	81
21651	100	01	82
21720	100	01	83
24101	100	01	84
24202	100	01	85
24221	100	01	86
24322	100	01	87
24334	100	01	88

24351	100	01	89
24370	100	01	90
24424	100	01	91
24425	100	01	92
24451	100	01	93
24626	100	02	81
24652	100	02	82
24683	100	02	83
24688	100	02	84
24691	100	02	85
24704	100	02	86
24711	100	02	87
24718	100	02	88
24722	100	02	89
24740	100	02	90
24771	100	02	91
24774	100	02	92
82101	100	02	93
82102	100	03	81
82103	100	03	82
82104	100	03	83
82111	100	03	84
82121	100	03	85
82122	100	03	86
82141	100	03	87
82142	100	03	88
82143	100	03	89
82171	100	03	90
82172	100	03	91
82173	100	03	92
82174	100	03	93
82201	100	04	81
82208	100	04	82
82211	100	04	83
82241	100	04	84
82242	100	04	85
82271	100	04	86
82281	100	04	87
82283	100	04	88
82291	100	04	89
82311	100	04	90
82342	100	04	91
82343	100	04	92
82345	100	04	93
82373	100	05	81
82411	100	05	82
82425	100	05	83
82444	100	05	84
82455	100	05	85
57149	100	05	86
57151	100	05	87
57152	100	05	88
57173	100	05	89
57284	100	05	90
57480	100	05	91
57780	100	05	92
80180	100	05	93
80211	100	06	81
80212	100	06	82
80223	100	06	83
80282	100	06	84
80327	100	06	85
80381	100	06	86
80627	100	06	87
80681	100	06	88
33115	100	06	89
33121	100	06	90
33122	100	06	91
33124	100	06	92
33141	100	06	93

33142	100	07	81
33151	100	07	82
33211	100	07	83
33231	100	07	84
33331	100	07	85
33338	100	07	86
33341	100	07	87
33441	100	07	88
33445	100	07	89
33650	100	07	90
33755	100	07	91
33759	100	07	92
33761	100	07	93
33778	100	08	81
33779	100	08	82
33783	100	08	83
85102	100	08	84
85211	100	08	85
85213	100	08	86
85219	100	08	87
85241	100	08	88
85320	100	08	89
85340	100	08	90
85341	100	08	91
85370	100	08	92
85390	100	08	93
85408	100	09	81
85414	100	09	82
85484	100	09	83
85738	100	09	84
85744	100	18	85
85765	100	09	86
85770	100	09	87
16161	100	09	88
16384	100	09	89
16456	100	09	90
16722	100	09	91
16811	100	09	92
16822	100	09	93
88150	100	10	81
88230	100	10	82
88251	100	10	83
88302	100	10	84
88341	100	10	85
88411	100	10	86
17651	100	10	87
36200	100	10	88
36201	100	10	89
36202	100	10	90
36208	100	10	91
36217	100	10	92
36220	100	10	93
36225	100	11	81
36309	100	11	82
36461	100	11	83
36661	100	11	84
36700	100	11	85
36705	100	11	86
36707	100	11	87
36749	100	11	88

B. Student ID and Their Final Schedule

[0, [325, 355, 367, 424, 442]], [1, [330, 331, 394, 421, 454]], [2, [4, 34, 236, 254, 265]], [3, [137, 240, 284, 296, 307]], [4, [30, 261, 295, 355, 438]], [5, [228, 316, 370, 386]], [6, [116, 228, 408, 454]], [7, [181, 374, 395, 399, 429]], [8, [183, 242, 333, 374, 380]], [9, [137, 139, 164, 282, 298]], [10, [121, 124, 143, 239, 342]], [11, [170, 297, 317, 353, 357]], [12, [135, 171, 459]], [13, [154, 352, 360, 368, 378]], [14, [168, 302, 336, 435, 446]], [15, [126, 209, 391, 427, 447]], [16, [215, 232, 392, 405, 431]], [17, [168, 307, 312, 347]], [18, [72, 311, 361, 427, 460]], [19, [19, 135, 165, 241, 275]], [20, [158, 382, 393, 401, 433]], [21, [267, 353, 365, 387, 402]], [22, [87, 199, 241, 355, 418]], [23, [108, 382]], [24, [162, 327, 363, 380, 391]], [25, [123, 299, 408, 436]], [26, [82, 294, 416, 456]], [27, [164, 226, 249, 342, 392]], [28, [110, 140, 416, 431, 434]], [29, [240, 375, 407, 415, 458]], [30, [354, 383, 389, 395, 440]], [31, [327, 396, 462]], [32, [62, 83, 152, 157, 246]], [33, [119, 171, 287, 393, 443]], [34, [98, 116, 277, 372, 400]], [35, [171, 208, 397, 420, 451]], [36, [203, 260, 343, 356, 363]], [37, [207, 288, 369, 386, 387]], [38, [168, 245, 334, 415, 446]], [39, [7, 40, 99, 239, 263]], [40, [218, 334, 381, 398, 414]], [41, [115, 131, 339, 371, 399]], [42, [115, 124, 282, 325, 348]], [43, [131, 268, 414, 443]], [44, [116]], [45, [96, 126, 323, 369, 389]], [46, [97, 341, 409, 434, 443]], [47, [182, 399, 414, 423, 440]], [48, [387, 396, 417]], [49, [136, 347, 425, 435, 437]], [50, [221, 245, 338, 389, 451]], [51, [69, 223, 327, 428, 431]], [52, [37, 204, 225, 304, 418]], [53, [42, 114, 129, 406]], [54, [119, 322, 394]], [55, [313, 332, 349, 419, 446]], [56, [58, 95, 409, 417, 449]], [57, [30, 99, 403, 435, 457]], [58, [287, 422, 435, 447]], [59, [135, 242, 334, 346, 406]], [60, [119, 134, 230, 327, 374]], [61, [14, 129, 229, 274, 328]], [62, [321, 368, 444]], [63, [235, 368, 455]], [64, [57, 264, 409, 430, 454]], [65, [402, 458]], [66, [10, 314, 386, 399]], [67, [43, 50, 356, 388, 435]], [68, [33, 94, 326, 381]], [69, [198, 332, 441]], [70, [197, 260, 352, 419, 422]], [71, [13, 300, 313, 442]], [72, [424, 448]], [73, [84, 146, 255, 312, 335]], [74, [29, 439, 454]], [75, [196, 208, 322, 412, 446]], [76, [224, 330, 348, 388, 435]], [77, [65, 117, 213, 297, 368]], [78, [93, 360, 381, 417, 457]], [79, [3, 92, 137, 204, 225]], [80, [1, 217, 237, 320, 410]], [81, [120, 197, 199, 332, 424]], [82, [314, 327, 382, 403, 417]], [83, [404, 438]], [84, [266, 345, 346, 403, 456]], [85, [212, 289, 325, 343, 395]], [86, [93, 245, 387, 432, 451]], [87, [93, 168, 339, 360, 363]], [88, [11, 313, 384, 403]], [89, [272, 351, 364, 450, 462]], [90, [139, 168, 175, 287, 293]], [91, [138, 168, 170, 254, 387]], [92, [181, 332, 336, 389, 397]], [93, [331, 386, 417]], [94, [313, 361, 384, 423, 457]], [95, [176, 383, 450]], [96, [174, 305, 340, 395, 439]], [97, [220, 397, 429, 443]], [98, [198, 250, 328, 368, 391]], [99, [328, 408, 418, 435, 447]], [100, [155, 172, 274, 410, 442]], [101, [201, 393, 397]], [102, [13, 62, 102, 104, 163]], [103, [92, 214, 290, 340, 410]], [104, [264, 373, 406, 437]], [105, [243, 366, 427, 451, 457]], [106, [109, 227, 286, 330, 358]], [107, [258, 337, 346, 420, 432]], [108, [148, 299, 346, 372, 427]], [109, [111, 166, 187, 333, 353]], [110, [232, 326, 337, 379, 404]], [111, [346, 359, 456]], [112, [413, 451, 453]], [113, [205, 364, 391, 452, 456]], [114, [229, 407, 416, 441]], [115, [236]], [116, [41, 230, 251, 291, 326]], [117, [55, 208, 249, 278, 399]], [118, [141, 271, 371, 379]], [119, [55, 393]],

[120, [305, 311, 354, 361, 410]], [121, [6, 208, 211, 295, 362]], [122, [238, 279, 333, 377, 394]], [123, [157, 325, 363, 413, 443]], [124, [29, 92, 220, 330, 406]], [125, [114, 136, 345, 356, 406]], [126, [95, 141, 252, 331, 428]], [127, [158, 209, 313, 350, 359]], [128, [245, 315, 428, 454]], [129, [121, 195, 278, 351, 361]], [130, [325, 365]], [131, [309, 355, 376]], [132, [115, 122, 125, 270, 326]], [133, [94, 99, 124, 128, 140]], [134, [321, 357, 406, 417, 418]], [135, [109, 161, 233, 403, 419]], [136, [238, 414, 455, 459]], [137, [140, 384, 417, 440]], [138, [57, 329, 390, 438]], [139, [272, 329, 345, 370, 373]], [140, [399, 414]], [141, [307, 367, 403, 420, 457]], [142, [97, 341, 457]], [143, [183, 207, 318, 338, 380]], [144, [20, 45, 153, 254, 375]], [145, [134, 223, 228, 262, 299]], [146, [189, 220, 260, 331, 356]], [147, [69, 324, 341, 350]], [148, [119, 237, 287, 307, 351]], [149, [144, 172, 179, 185, 320]], [150, [101, 143, 212, 361, 408]], [151, [21, 130, 222, 380, 384]], [152, [40, 313, 383, 414, 418]], [153, [265, 327, 357, 376, 426]], [154, [71, 274, 342, 357, 360]], [155, [135, 225, 365, 398, 415]], [156, [177, 338, 359, 427, 462]], [157, [285, 341, 410, 419, 461]], [158, [150, 406, 435, 445, 446]], [159, [346, 446, 459]], [160, [231, 314, 338, 342, 365]], [161, [40, 219, 273, 284, 336]], [162, [157, 189, 332, 429, 460]], [163, [132, 328, 417, 458]], [164, [15, 183, 238, 347, 350]], [165, [106, 160, 255, 256, 379]], [166, [415]], [167, [365, 400, 424, 447, 453]], [168, [128, 359]], [169, [26, 279, 412, 432, 433]], [170, [180, 286, 312, 376, 378]], [171, [207, 233, 307, 376, 428]],.....

The other 5930 students' schedules are omitted due to the length of data.

C. Classroom Information

ROOM	SEATS		
Baker Hall		**Classroom by size**	
A51 (Giant Eagl	144	<50	70
A53 (Steinberg)	73	50-100	10
A54 (Osher)	15	>100	13
136A (Adamson)	111	Total	93
140 C	20		
140 D	10		
140 E	30		
140 F	30		
154A	12		
235A	35		

235B	35			
237B	35			
255A	35			
College of Fine Arts				
102	35			
317	20			
318	20			
323	10			
321	2			
Doherty Hall				
A302	132			
1112	98			
1117	35			
1209	30			
1211	35			
1212	107			
1217	35			
2105	35			
2122	35			
2210	289			
2302	113			
2315	266			
Gates Hillman Center				
4101	26			
4102	40			
4211	42			
4215	58			
4301	28			
4307	73			
4401	244			
5222	38			
Hamerschlag Hall				

B103	101			
B131	98			
Hunt Library				
Near	25			
Far	25			
Near & Far	50			
Margaret Morrison				
A14	108			
103	112			
Mellon Institute				
348	60			
355	25			
448	25			
Porter Hall				
A18A	50			
A18B	50			
A18C	50			
A19A	20			
A19B	20			
A19C	18			
A19D	18			
A20A	21			
A21	21			
A21A	21			
A22	30			
100	217			
125B	28			
125C	70			
126A	28			
225B	28			
226A	28			
226B	28			

226C	28			
Scaife Hall				
125	96			
208	30			
212	20			
214	45			
219	45			
220	35			
222	35			
Wean Hall				
4623	45			
4709	35			
5201	30			
5202	30			
5207	20			
5302	40			
5304	20			
5310	30			
5312	30			
5316	20			
5320	30			
5328	24			
5403	65			
5409	55			
5415	45			
5421	45			
6423	30			
7500	152			
8427	30			

D. Code

```
# Data format for each student: [id, numberOfClass, boolForClass]
import random
import copy
def initiate_ref(num):
        rst = []
        for i in xrange(num):
                rst += [[i, 0]]
        return rst

def initiate_student(num):
        rst = []
        for i in xrange(num):
                rst += [[i, []]]
        return rst

def initiateClass(studentList, takerNum, seed, rstList, classLable):
        random.seed(seed)

        availableList = []
        for student in studentList:
                student += [False]
                if student[1] != 5:
                        availableList += [student[0]]
        if len(availableList) >= takerNum:
                takers = random.sample(availableList, takerNum)
                for taker in takers:
                        studentList[taker][-1] = True
                        studentList[taker][1] += 1
                        rstList[taker][1]+=[classLable]
                return True
        else:
                for taker in availableList:
                        studentList[taker][-1] = True
                        studentList[taker][1] += 1
                        rstList[taker][1] += [classLable]
                return False

def getList(num_studnet, classList, seed):
        studentList = initiate_ref(num_studnet)
        rst_list = initiate_student(num_studnet)
        # print len(classList)
        for index in xrange(len(classList)):
                num_takers = classList[index]
                seed += 1
                initiateClass(studentList, num_takers[1], seed, rst_list,
index)

        return rst_list

def printList(studnetList):
        for student in studentList:
                print str(studnet[0]) + "   " + str(student[1])
```

```
def readClassList(filename):
        file = open(filename, "r")
        rst = []
        for line in file:
                crt_class_str = int(line[0:5])
                crt_class_num = int(crt_class_str)
                crt_student_num = int(line[6:10])
                rst += [(crt_class_num, crt_student_num)]
        return rst

def readSchedule(filename):
        file = open(filename, "r")
        rst = []
        for line in file:
                crt_class_str = int(line[0:5])

                crt_class_num = int(crt_class_str)
                crt_student_num = int(line[6:10])
                crt_time = int(line[10:13])
                crt_room = int(line[13:])
                rst += [(crt_class_num, crt_student_num, crt_time, crt_room)]
        return rst




# print schedule
# print studentList

#(list: classlable, # students, time slot, classroom index)
# 1 - 70 are small classrooms, 71-80 are mid size classrooms, 81 - 93 are big
classrooms

# student bool list: [studentLable, number of class taking, bools...]
#
def findSame(timeList, countList):
        num_conflict = 0
        count = []
        for i in xrange(len(timeList)):
                count += [False]
        if len(timeList) < 2:
                return 0
        for i in xrange(len(timeList)-1):
                if timeList[i][0] == timeList[i+1][0]:
                        if count[i] == False:
                                countList[timeList[i][1]] += 1
                                countList[timeList[i+1][1]] += 1
                        count[i] = True
                        num_conflict += 1
        return num_conflict


def findContinuous(timeList, countList):
        num_conflict = 0
        # print len(timeList)
```

```
        if len(timeList) < 3:
                return 0
        for i in xrange(len(timeList)-2):
                if timeList[i][0] == timeList[i+1][0]-1 and timeList[i][0] ==
timeList[i+2][0]-2:
                        countList[timeList[i][1]] += 1
                        countList[timeList[i+1][1]] += 1
                        countList[timeList[i+2][1]] += 1
                        num_conflict += 1
        return num_conflict

def findConflict(timeList):
        cont_3 = findContinuous(timeList)
        same_time = findSame(timeList)
        return (cont_3 + same_time)
# print studentList

def compare(a, b):
        return a[0] - b[0]

def sortCountList(countList, inputSchedule):
        returnedList = []
        for i in xrange(countList):
                returnedList += [(countList[i], i, inputSchedule[i])]
        list.sort(returnedList, cmp = compare)
        returnedSchedule = []
        for i in xrange(returnedList):
                returnedSchedule += [(returnedList[i][2][0],
returnedList[i][2][1], returnedList[i][2][0], i%18)]
        return returnedList




def calculateConflicts_maxtwo(scheduleList, studentList):
        cont = 0
        same = 0
        sameCountList = []

        studentConflictList = []
        changeList = []
        for i in xrange(len(scheduleList)):
                sameCountList += [0]
        contCountList = copy.deepcopy(sameCountList)
        for student in studentList:
                classList = student[1]
                # print classList
                timeList = []
                time_class_list = []

                for each_class_index in classList:
                        timeList += [scheduleList[each_class_index][2]]
                        time_class_list += [[scheduleList[each_class_index][2],
each_class_index]]
                list.sort(time_class_list, cmp = compare)
                list.sort(timeList)
```

```
                crt_same = findSame(time_class_list, sameCountList)
                same += crt_same
                cont += findContinuous(time_class_list, contCountList)
                studentConflictList += [crt_same]



        maxIndex = sameCountList.index(max(sameCountList))
        maxTime = scheduleList[maxIndex][2]


        sameCountList[maxIndex] = -1


        secondIndex = sameCountList.index(max(sameCountList))
        secondTime = scheduleList[secondIndex][2]
        sameCountList[secondIndex] = -1

        # loot for the biggest 2 conflicts and switch
        while (secondTime == maxTime):
                secondIndex = sameCountList.index(max(sameCountList))
                secondTime = scheduleList[secondIndex][2]
                sameCountList[secondIndex] = -1
        changeList += [(maxIndex, secondTime), (secondIndex, maxTime)]
        return changeList


def calculateConflicts_maxone(scheduleList, studentList):
        cont = 0
        same = 0
        sameCountList = []

        studentConflictList = []
        changeList = []
        for i in xrange(len(scheduleList)):
                sameCountList += [0]
        contCountList = copy.deepcopy(sameCountList)
        for student in studentList:
                classList = student[1]
                # print classList
                timeList = []
                time_class_list = []

                for each_class_index in classList:
                        timeList += [scheduleList[each_class_index][2]]
                        time_class_list += [[scheduleList[each_class_index][2],
each_class_index]]
                list.sort(time_class_list, cmp = compare)
                list.sort(timeList)
                crt_same = findSame(time_class_list, sameCountList)
                same += crt_same
                cont += findContinuous(time_class_list, contCountList)
                studentConflictList += [crt_same]



        maxIndex = sameCountList.index(max(sameCountList))
```

```
        maxTime = scheduleList[maxIndex][2]


        sameCountList[maxIndex] = -1


        secondIndex = sameCountList.index(max(sameCountList))
        secondTime = scheduleList[secondIndex][2]
        sameCountList[secondIndex] = -1

        # loot for the biggest 2 conflicts and switch
        while (secondTime == maxTime):
                secondIndex = sameCountList.index(max(sameCountList))
                secondTime = scheduleList[secondIndex][2]
                sameCountList[secondIndex] = -1

        changeList += [(maxIndex, secondTime), (secondIndex, maxTime)]

        return changeList


def changeSchedule(changeList, schedule):
        newschedule = copy.deepcopy(schedule)
        for i in xrange(len(changeList)):
                classIndex = changeList[i][0]
                time = changeList[i][1]
                newschedule[classIndex] = (schedule[classIndex][0],
schedule[classIndex][1], time, schedule[classIndex][3])
        return newschedule



def loopForLargest(inputSchedule, studentList, loop_num):
        crt_schedule = inputSchedule
        for i in xrange(loop_num):
                changeList = calculateConflicts_maxtwo(crt_schedule,
studentList)
                crt_schedule = changeSchedule(changeList, crt_schedule)
        return crt_schedule



def calculateConflicts_maxinf(scheduleList, studentList, swapOffSet):
        cont = 0
        same = 0
        sameCountList = []

        studentConflictList = []
        changeList = []
        for i in xrange(len(scheduleList)):
                sameCountList += [0]
        contCountList = copy.deepcopy(sameCountList)
        for student in studentList:
                classList = student[1]
                # print classList
                timeList = []
                time_class_list = []
```

```
                for each_class_index in classList:
                        timeList += [scheduleList[each_class_index][2]]
                        time_class_list += [[scheduleList[each_class_index][2],
each_class_index]]
                list.sort(time_class_list, cmp = compare)
                list.sort(timeList)
                crt_same = findSame(time_class_list, sameCountList)
                same += crt_same
                cont += findContinuous(time_class_list, contCountList)
                studentConflictList += [crt_same]
        # print "max"
        print max(sameCountList)

        maxIndex = sameCountList.index(max(sameCountList))
        maxTime = scheduleList[maxIndex][2]


        sameCountList[maxIndex] = -1


        secondIndex = sameCountList.index(max(sameCountList))
        secondTime = scheduleList[secondIndex][2]
        sameCountList[secondIndex] = -1
        thirdTime = secondTime

        # loot for the biggest 2 conflicts and switch
        while (secondTime == maxTime):
                secondIndex = sameCountList.index(max(sameCountList))
                secondTime = scheduleList[secondIndex][2]
                sameCountList[secondIndex] = -1
        for i in xrange(swapOffSet):
                thirdIndex = sameCountList.index(max(sameCountList))
                thirdTime = scheduleList[thirdIndex][2]
                sameCountList[secondIndex] = -1
        while (thirdTime == maxTime):
                thirdIndex = sameCountList.index(max(sameCountList))
                thirdTime = scheduleList[thirdIndex][2]
                sameCountList[thirdIndex] = -1


        if swapOffSet != 0:
                changeList += [(thirdIndex, maxTime), (maxIndex, thirdTime)]
        else:
                changeList += [(maxIndex, secondTime), (secondIndex, maxTime)]
        print same
        return [same, changeList]



def calculateConflicts_maxinf_breakTwo(scheduleList, studentList, swapOffSet,
BreakTwo):
        cont = 0
        same = 0
        sameCountList = []
```

```
        studentConflictList = []
        changeList = []
        for i in xrange(len(scheduleList)):
                sameCountList += [0]
        contCountList = copy.deepcopy(sameCountList)
        for student in studentList:
                classList = student[1]
                timeList = []
                time_class_list = []
                for each_class_index in classList:
                        timeList += [scheduleList[each_class_index][2]]
                        time_class_list += [[scheduleList[each_class_index][2],
each_class_index]]
                list.sort(time_class_list, cmp = compare)
                list.sort(timeList)
                crt_same = findSame(time_class_list, sameCountList)
                same += crt_same
                cont += findContinuous(time_class_list, contCountList)
                studentConflictList += [crt_same]
        maxIndex = sameCountList.index(max(sameCountList))
        maxTime = scheduleList[maxIndex][2]


        sameCountList[maxIndex] = -1


        secondIndex = sameCountList.index(max(sameCountList))
        secondTime = scheduleList[secondIndex][2]
        sameCountList[secondIndex] = -1
        thirdTime = secondTime

        # loot for the biggest 2 conflicts and switch
        while (secondTime == maxTime):
                secondIndex = sameCountList.index(max(sameCountList))
                secondTime = scheduleList[secondIndex][2]
                sameCountList[secondIndex] = -1
        for i in xrange(swapOffSet):
                thirdIndex = sameCountList.index(max(sameCountList))
                thirdTime = scheduleList[thirdIndex][2]
                sameCountList[thirdIndex] = -1
        while (thirdTime == maxTime):
                thirdIndex = sameCountList.index(max(sameCountList))
                thirdTime = scheduleList[thirdIndex][2]
                sameCountList[thirdIndex] = -1


        if swapOffSet != 0:
                if BreakTwo:
                        changeList += [(secondIndex, thirdTime), (thirdIndex,
secondTime)]
                else:
                        changeList += [(thirdIndex, maxTime), (maxIndex,
thirdTime)]
        else:
                changeList += [(maxIndex, secondTime), (secondIndex, maxTime)]
        print same
        return [same, changeList]
```

```
def loopForLargest_inf(inputSchedule, studentList, loop_num):
	crt_schedule = inputSchedule
	crt_conflict = 1000000
	for i in xrange(loop_num):
		offset = 1
		temp = calculateConflicts_maxinf(crt_schedule, studentList, 0)
		conflict = temp[0]
		changeList = temp[1]
		if conflict < crt_conflict or conflict > crt_conflict:
			crt_schedule = changeSchedule(changeList, crt_schedule)
		elif conflict == crt_conflict:
			while conflict >= crt_conflict:
				temp = calculateConflicts_maxinf(crt_schedule,
studentList, offset)
				offset = offset + 1
				conflict = temp[0]
				changeList = temp[1]
				crt_schedule = changeSchedule(changeList,
crt_schedule)
		crt_conflict = conflict
	return crt_schedule


def comp_changeList(x, y):
	a, b = x[0], x[1]
	c, d = y[0], y[1]

	if (a == c and b == d) or (b == c and a == d):
		return True
	else:
		return False

def loopForLargest_inf_breakTwo(inputSchedule, studentList, loop_num):
	crt_schedule = inputSchedule
	crt_conflict = 1000000
	conflictList = []
	for i in xrange(loop_num):
		offset = 1
		temp = calculateConflicts_maxinf_breakTwo(crt_schedule,
studentList, 0, False)
		conflict = temp[0]
		changeList = temp[1]
		old_change_1 = [changeList[0][0], changeList[1][0]]
		old_change_2 = old_change_1
		if conflict < crt_conflict:
			crt_schedule = changeSchedule(changeList, crt_schedule)
		elif conflict >= crt_conflict:
			while abs(conflict - crt_conflict) <=15 or
comp_changeList([changeList[0][0], changeList[1][0]], old_change_2):
				temp =
calculateConflicts_maxinf_breakTwo(crt_schedule, studentList, offset, False)
				offset = offset + 1
				conflict = temp[0]
				changeList = temp[1]
```

```
                                if comp_changeList([changeList[0][0],
changeList[1][0]], old_change_2):
                                    temp =
calculateConflicts_maxinf_breakTwo(crt_schedule, studentList, offset, True)
                                    conflict = temp[0]
                                    changeList = temp[1]
                                old_change_2 = old_change_1
                                old_change_1 = [changeList[0][0],
changeList[1][0]]
                                crt_schedule = changeSchedule(changeList,
crt_schedule)
                conflictList += [conflict]
                crt_conflict = conflict
        return (conflictList, crt_schedule)



class_num = 465
classStudent_num = readClassList("new.txt")
studentList = getList(6100, classStudent_num, 1)
schedule = readSchedule("new.txt")
result = loopForLargest_inf_breakTwo(schedule, studentList, 1000)
conflictdata = result[0]
scheduleData = result[1]
```

E. Optimal Schedule

Course	Max Enroll	Date	Time
48250	20	Monday, December 11, 2017	8:30-11:30 a.m.
48315	20	Monday, December 11, 2017	1:00-4:00 p.m.
48324	20	Monday, December 11, 2017	5:30-8:30 p.m.
48357	20	Tuesday, December 12, 2017	8:30-11:30 a.m.
48524	20	Tuesday, December 12, 2017	1:00-4:00 p.m.
48558	20	Tuesday, December 12, 2017	5:30-8:30 p.m.
48568	20	Thursday, December 14, 2017	8:30-11:30 a.m.
48635	20	Thursday, December 14, 2017	1:00-4:00 p.m.
48729	20	Thursday, December 14, 2017	5:30-8:30 p.m.
48734	20	Friday, December 15, 2017	8:30-11:30 a.m.
48740	20	Friday, December 15, 2017	1:00-4:00 p.m.
48771	20	Friday, December 15, 2017	5:30-8:30 p.m.
60157	20	Sunday, December 17, 2017	8:30-11:30 a.m.
60220	20	Sunday, December 17, 2017	1:00-4:00 p.m.
60424	20	Sunday, December 17, 2017	5:30-8:30 p.m.
3121	20	Monday, December 18, 2017	8:30-11:30 a.m.

3124	20	Monday, December 18, 2017	1:00-4:00 p.m.
3125	20	Monday, December 18, 2017	5:30-8:30 p.m.
3133	20	Monday, December 11, 2017	8:30-11:30 a.m.
3151	20	Monday, December 11, 2017	1:00-4:00 p.m.
3201	20	Monday, December 11, 2017	5:30-8:30 p.m.
30220	20	Tuesday, December 12, 2017	8:30-11:30 a.m.
30301	20	Tuesday, December 12, 2017	1:00-4:00 p.m.
3320	20	Tuesday, December 12, 2017	5:30-8:30 p.m.
3327	20	Thursday, December 14, 2017	8:30-11:30 a.m.
3362	20	Thursday, December 14, 2017	1:00-4:00 p.m.
3401	20	Thursday, December 14, 2017	5:30-8:30 p.m.
3439	20	Friday, December 15, 2017	8:30-11:30 a.m.
3511	20	Friday, December 15, 2017	1:00-4:00 p.m.
3534	20	Friday, December 15, 2017	5:30-8:30 p.m.
3709	20	Sunday, December 17, 2017	8:30-11:30 a.m.
3711	20	Sunday, December 17, 2017	1:00-4:00 p.m.
3727	20	Sunday, December 17, 2017	5:30-8:30 p.m.
42101	20	Monday, December 18, 2017	8:30-11:30 a.m.
42202	20	Monday, December 18, 2017	1:00-4:00 p.m.
42302	20	Monday, December 18, 2017	5:30-8:30 p.m.
42341	20	Monday, December 11, 2017	8:30-11:30 a.m.
42401	20	Monday, December 11, 2017	1:00-4:00 p.m.
42620	20	Monday, December 11, 2017	5:30-8:30 p.m.
42671	20	Tuesday, December 12, 2017	8:30-11:30 a.m.
42674	20	Tuesday, December 12, 2017	1:00-4:00 p.m.
42773	20	Tuesday, December 12, 2017	5:30-8:30 p.m.
70122	20	Thursday, December 14, 2017	8:30-11:30 a.m.
70208	20	Thursday, December 14, 2017	1:00-4:00 p.m.
70311	20	Thursday, December 14, 2017	5:30-8:30 p.m.
70318	20	Friday, December 15, 2017	8:30-11:30 a.m.
70341	20	Friday, December 15, 2017	1:00-4:00 p.m.
70365	20	Friday, December 15, 2017	5:30-8:30 p.m.
70391	20	Sunday, December 17, 2017	8:30-11:30 a.m.
70398	20	Sunday, December 17, 2017	1:00-4:00 p.m.
70451	20	Sunday, December 17, 2017	5:30-8:30 p.m.
70460	20	Monday, December 18, 2017	8:30-11:30 a.m.
70477	20	Monday, December 18, 2017	1:00-4:00 p.m.
70497	20	Monday, December 18, 2017	5:30-8:30 p.m.
52400	20	Monday, December 11, 2017	8:30-11:30 a.m.

52401	20	Monday, December 11, 2017	1:00-4:00 p.m.
62110	20	Monday, December 11, 2017	5:30-8:30 p.m.
62150	20	Tuesday, December 12, 2017	8:30-11:30 a.m.
62633	20	Tuesday, December 12, 2017	1:00-4:00 p.m.
6100	20	Tuesday, December 12, 2017	5:30-8:30 p.m.
6221	20	Thursday, December 14, 2017	8:30-11:30 a.m.
6321	20	Thursday, December 14, 2017	1:00-4:00 p.m.
6323	20	Thursday, December 14, 2017	5:30-8:30 p.m.
6423	20	Friday, December 15, 2017	8:30-11:30 a.m.
6614	20	Friday, December 15, 2017	1:00-4:00 p.m.
6623	20	Friday, December 15, 2017	5:30-8:30 p.m.
6703	20	Sunday, December 17, 2017	8:30-11:30 a.m.
6705	20	Sunday, December 17, 2017	1:00-4:00 p.m.
6815	20	Sunday, December 17, 2017	5:30-8:30 p.m.
9105	20	Monday, December 18, 2017	8:30-11:30 a.m.
9106	20	Monday, December 18, 2017	1:00-4:00 p.m.
9107	20	Monday, December 18, 2017	5:30-8:30 p.m.
9207	20	Monday, December 11, 2017	8:30-11:30 a.m.
9214	20	Monday, December 11, 2017	1:00-4:00 p.m.
9217	20	Monday, December 11, 2017	5:30-8:30 p.m.
9219	20	Tuesday, December 12, 2017	8:30-11:30 a.m.
9221	20	Tuesday, December 12, 2017	1:00-4:00 p.m.
9231	20	Tuesday, December 12, 2017	5:30-8:30 p.m.
9344	20	Thursday, December 14, 2017	8:30-11:30 a.m.
9347	20	Thursday, December 14, 2017	1:00-4:00 p.m.
9536	20	Thursday, December 14, 2017	5:30-8:30 p.m.
9721	20	Friday, December 15, 2017	8:30-11:30 a.m.
9736	20	Friday, December 15, 2017	1:00-4:00 p.m.
39402	20	Friday, December 15, 2017	5:30-8:30 p.m.
39613	20	Sunday, December 17, 2017	8:30-11:30 a.m.
12100	20	Sunday, December 17, 2017	1:00-4:00 p.m.
12335	20	Sunday, December 17, 2017	5:30-8:30 p.m.
12355	20	Monday, December 18, 2017	8:30-11:30 a.m.
12411	20	Monday, December 18, 2017	1:00-4:00 p.m.
12620	20	Monday, December 18, 2017	5:30-8:30 p.m.
12629	20	Monday, December 11, 2017	8:30-11:30 a.m.
12635	20	Monday, December 11, 2017	1:00-4:00 p.m.
12651	20	Monday, December 11, 2017	5:30-8:30 p.m.
12702	50	Sunday, December 17, 2017	5:30-8:30 p.m.

12704	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
12709	50	Tuesday, December 12, 2017	5:30-8:30 p.m.
12712	50	Thursday, December 14, 2017	8:30-11:30 a.m.
12720	50	Thursday, December 14, 2017	1:00-4:00 p.m.
12729	50	Monday, December 11, 2017	5:30-8:30 p.m.
12741	50	Friday, December 15, 2017	8:30-11:30 a.m.
12751	50	Friday, December 15, 2017	1:00-4:00 p.m.
12752	50	Friday, December 15, 2017	8:30-11:30 a.m.
2261	50	Sunday, December 17, 2017	8:30-11:30 a.m.
2601	50	Sunday, December 17, 2017	1:00-4:00 p.m.
2613	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
2711	50	Monday, December 18, 2017	8:30-11:30 a.m.
15110	50	Monday, December 18, 2017	1:00-4:00 p.m.
15112	50	Monday, December 18, 2017	5:30-8:30 p.m.
15121	50	Monday, December 11, 2017	8:30-11:30 a.m.
15122	50	Monday, December 11, 2017	1:00-4:00 p.m.
15150	50	Monday, December 11, 2017	5:30-8:30 p.m.
15151	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
15210	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
15214	50	Tuesday, December 12, 2017	5:30-8:30 p.m.
15251	50	Thursday, December 14, 2017	8:30-11:30 a.m.
15313	50	Thursday, December 14, 2017	1:00-4:00 p.m.
15317	50	Thursday, December 14, 2017	5:30-8:30 p.m.
15322	50	Friday, December 15, 2017	8:30-11:30 a.m.
15351	50	Friday, December 15, 2017	1:00-4:00 p.m.
15381	50	Friday, December 15, 2017	5:30-8:30 p.m.
15410	50	Sunday, December 17, 2017	8:30-11:30 a.m.
15414	50	Sunday, December 17, 2017	1:00-4:00 p.m.
15421	50	Sunday, December 17, 2017	5:30-8:30 p.m.
15437	50	Monday, December 18, 2017	8:30-11:30 a.m.
15440	50	Monday, December 18, 2017	1:00-4:00 p.m.
15441	50	Monday, December 18, 2017	5:30-8:30 p.m.
15445	50	Monday, December 11, 2017	8:30-11:30 a.m.
15451	50	Monday, December 11, 2017	1:00-4:00 p.m.
15456	50	Monday, December 11, 2017	5:30-8:30 p.m.
15458	50	Monday, December 11, 2017	8:30-11:30 a.m.
15462	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
15463	50	Tuesday, December 12, 2017	5:30-8:30 p.m.
15487	50	Thursday, December 14, 2017	8:30-11:30 a.m.

15605	50	Thursday, December 14, 2017	1:00-4:00 p.m.
15614	50	Monday, December 11, 2017	8:30-11:30 a.m.
15622	50	Friday, December 15, 2017	8:30-11:30 a.m.
15637	50	Friday, December 15, 2017	1:00-4:00 p.m.
15640	50	Friday, December 15, 2017	5:30-8:30 p.m.
15641	50	Friday, December 15, 2017	1:00-4:00 p.m.
15645	50	Sunday, December 17, 2017	1:00-4:00 p.m.
15650	50	Sunday, December 17, 2017	5:30-8:30 p.m.
15651	50	Monday, December 18, 2017	8:30-11:30 a.m.
15657	50	Monday, December 18, 2017	1:00-4:00 p.m.
15662	50	Monday, December 18, 2017	5:30-8:30 p.m.
15663	50	Monday, December 11, 2017	8:30-11:30 a.m.
15681	50	Monday, December 11, 2017	1:00-4:00 p.m.
15852	50	Friday, December 15, 2017	8:30-11:30 a.m.
15858	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
15862	50	Monday, December 11, 2017	1:00-4:00 p.m.
15883	50	Thursday, December 14, 2017	1:00-4:00 p.m.
66161	50	Thursday, December 14, 2017	8:30-11:30 a.m.
66221	50	Thursday, December 14, 2017	1:00-4:00 p.m.
67240	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
67262	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
67328	50	Friday, December 15, 2017	1:00-4:00 p.m.
67329	50	Thursday, December 14, 2017	1:00-4:00 p.m.
73102	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
73230	50	Sunday, December 17, 2017	1:00-4:00 p.m.
73240	50	Thursday, December 14, 2017	1:00-4:00 p.m.
73328	50	Monday, December 18, 2017	8:30-11:30 a.m.
73347	50	Thursday, December 14, 2017	8:30-11:30 a.m.
73366	50	Monday, December 18, 2017	5:30-8:30 p.m.
73374	50	Monday, December 11, 2017	8:30-11:30 a.m.
73421	50	Monday, December 11, 2017	1:00-4:00 p.m.
18090	50	Monday, December 11, 2017	5:30-8:30 p.m.
18202	50	Thursday, December 14, 2017	5:30-8:30 p.m.
18220	50	Tuesday, December 12, 2017	5:30-8:30 p.m.
18240	50	Friday, December 15, 2017	5:30-8:30 p.m.
18290	50	Thursday, December 14, 2017	8:30-11:30 a.m.
18300	50	Thursday, December 14, 2017	1:00-4:00 p.m.
18340	50	Thursday, December 14, 2017	5:30-8:30 p.m.
18349	50	Friday, December 15, 2017	8:30-11:30 a.m.

18370	50	Friday, December 15, 2017	1:00-4:00 p.m.
18372	50	Thursday, December 14, 2017	5:30-8:30 p.m.
18422	50	Monday, December 18, 2017	5:30-8:30 p.m.
18492	50	Sunday, December 17, 2017	1:00-4:00 p.m.
18618	50	Sunday, December 17, 2017	5:30-8:30 p.m.
18622	50	Monday, December 11, 2017	5:30-8:30 p.m.
18631	50	Monday, December 18, 2017	1:00-4:00 p.m.
18639	50	Monday, December 18, 2017	5:30-8:30 p.m.
18643	50	Monday, December 11, 2017	8:30-11:30 a.m.
18644	50	Monday, December 11, 2017	1:00-4:00 p.m.
18648	50	Monday, December 11, 2017	5:30-8:30 p.m.
18650	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
18715	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
18730	50	Tuesday, December 12, 2017	5:30-8:30 p.m.
18749	50	Thursday, December 14, 2017	8:30-11:30 a.m.
18751	50	Thursday, December 14, 2017	5:30-8:30 p.m.
18755	50	Sunday, December 17, 2017	8:30-11:30 a.m.
18765	50	Friday, December 15, 2017	8:30-11:30 a.m.
18771	50	Friday, December 15, 2017	1:00-4:00 p.m.
18781	50	Monday, December 18, 2017	1:00-4:00 p.m.
18785	50	Sunday, December 17, 2017	8:30-11:30 a.m.
18792	50	Sunday, December 17, 2017	1:00-4:00 p.m.
18793	50	Sunday, December 17, 2017	5:30-8:30 p.m.
18794	50	Monday, December 18, 2017	8:30-11:30 a.m.
18847	50	Monday, December 18, 2017	1:00-4:00 p.m.
19101	50	Sunday, December 17, 2017	5:30-8:30 p.m.
19301	50	Monday, December 11, 2017	8:30-11:30 a.m.
19403	50	Monday, December 11, 2017	1:00-4:00 p.m.
19424	50	Monday, December 11, 2017	5:30-8:30 p.m.
19440	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
19462	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
19472	50	Tuesday, December 12, 2017	5:30-8:30 p.m.
19638	50	Thursday, December 14, 2017	8:30-11:30 a.m.
19639	50	Thursday, December 14, 2017	1:00-4:00 p.m.
19691	50	Thursday, December 14, 2017	5:30-8:30 p.m.
19713	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
19717	50	Friday, December 15, 2017	1:00-4:00 p.m.
19740	50	Thursday, December 14, 2017	5:30-8:30 p.m.
19751	50	Monday, December 11, 2017	5:30-8:30 p.m.

76205	50	Sunday, December 17, 2017	1:00-4:00 p.m.
76223	50	Sunday, December 17, 2017	5:30-8:30 p.m.
76239	50	Monday, December 18, 2017	8:30-11:30 a.m.
76315	50	Monday, December 18, 2017	1:00-4:00 p.m.
76337	50	Sunday, December 17, 2017	5:30-8:30 p.m.
76366	50	Monday, December 11, 2017	8:30-11:30 a.m.
76375	50	Monday, December 11, 2017	1:00-4:00 p.m.
76389	50	Friday, December 15, 2017	8:30-11:30 a.m.
76396	50	Sunday, December 17, 2017	8:30-11:30 a.m.
76441	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
76450	50	Tuesday, December 12, 2017	5:30-8:30 p.m.
76457	50	Thursday, December 14, 2017	8:30-11:30 a.m.
76481	50	Monday, December 11, 2017	8:30-11:30 a.m.
76766	50	Thursday, December 14, 2017	5:30-8:30 p.m.
76775	50	Friday, December 15, 2017	8:30-11:30 a.m.
76789	50	Friday, December 15, 2017	1:00-4:00 p.m.
76791	50	Friday, December 15, 2017	5:30-8:30 p.m.
76796	50	Monday, December 11, 2017	5:30-8:30 p.m.
76841	50	Sunday, December 17, 2017	1:00-4:00 p.m.
76850	50	Sunday, December 17, 2017	5:30-8:30 p.m.
76857	50	Monday, December 18, 2017	8:30-11:30 a.m.
76881	50	Monday, December 18, 2017	1:00-4:00 p.m.
53353	50	Monday, December 18, 2017	5:30-8:30 p.m.
53451	50	Monday, December 11, 2017	8:30-11:30 a.m.
53740	50	Monday, December 11, 2017	1:00-4:00 p.m.
53751	50	Sunday, December 17, 2017	8:30-11:30 a.m.
79212	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
79229	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
79236	50	Thursday, December 14, 2017	5:30-8:30 p.m.
79259	50	Thursday, December 14, 2017	8:30-11:30 a.m.
79261	50	Thursday, December 14, 2017	1:00-4:00 p.m.
79265	50	Thursday, December 14, 2017	5:30-8:30 p.m.
79277	50	Friday, December 15, 2017	8:30-11:30 a.m.
79299	50	Friday, December 15, 2017	1:00-4:00 p.m.
79310	50	Friday, December 15, 2017	5:30-8:30 p.m.
79327	50	Friday, December 15, 2017	5:30-8:30 p.m.
79352	50	Sunday, December 17, 2017	1:00-4:00 p.m.
5341	50	Sunday, December 17, 2017	5:30-8:30 p.m.
5391	50	Monday, December 18, 2017	8:30-11:30 a.m.

5410	50	Monday, December 18, 2017	1:00-4:00 p.m.
5430	50	Friday, December 15, 2017	5:30-8:30 p.m.
5431	50	Monday, December 11, 2017	8:30-11:30 a.m.
5499	50	Monday, December 11, 2017	1:00-4:00 p.m.
5610	50	Monday, December 11, 2017	5:30-8:30 p.m.
5630	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
5631	50	Sunday, December 17, 2017	5:30-8:30 p.m.
5814	50	Tuesday, December 12, 2017	8:30-11:30 a.m.
5823	50	Thursday, December 14, 2017	8:30-11:30 a.m.
5891	50	Thursday, December 14, 2017	1:00-4:00 p.m.
5899	50	Sunday, December 17, 2017	8:30-11:30 a.m.
14642	50	Monday, December 11, 2017	1:00-4:00 p.m.
14741	50	Friday, December 15, 2017	1:00-4:00 p.m.
14848	50	Friday, December 15, 2017	8:30-11:30 a.m.
84326	50	Monday, December 11, 2017	1:00-4:00 p.m.
84369	50	Sunday, December 17, 2017	1:00-4:00 p.m.
84669	50	Sunday, December 17, 2017	5:30-8:30 p.m.
8631	50	Monday, December 18, 2017	8:30-11:30 a.m.
8672	50	Monday, December 18, 2017	1:00-4:00 p.m.
8722	50	Monday, December 18, 2017	5:30-8:30 p.m.
8736	50	Monday, December 11, 2017	8:30-11:30 a.m.
11291	50	Monday, December 11, 2017	1:00-4:00 p.m.
11411	50	Monday, December 11, 2017	5:30-8:30 p.m.
11492	50	Friday, December 15, 2017	5:30-8:30 p.m.
11611	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
11676	50	Tuesday, December 12, 2017	5:30-8:30 p.m.
11692	50	Thursday, December 14, 2017	8:30-11:30 a.m.
11711	50	Sunday, December 17, 2017	8:30-11:30 a.m.
11751	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
11777	50	Sunday, December 17, 2017	8:30-11:30 a.m.
11785	50	Friday, December 15, 2017	1:00-4:00 p.m.
11792	50	Friday, December 15, 2017	5:30-8:30 p.m.
11927	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
10601	50	Sunday, December 17, 2017	1:00-4:00 p.m.
10701	50	Sunday, December 17, 2017	5:30-8:30 p.m.
27100	50	Monday, December 18, 2017	8:30-11:30 a.m.
27201	50	Monday, December 18, 2017	1:00-4:00 p.m.
27215	50	Monday, December 18, 2017	5:30-8:30 p.m.
27301	50	Friday, December 15, 2017	5:30-8:30 p.m.

27324	50	Monday, December 11, 2017	1:00-4:00 p.m.
27432	50	Monday, December 11, 2017	5:30-8:30 p.m.
27502	50	Friday, December 15, 2017	5:30-8:30 p.m.
27766	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
27797	50	Tuesday, December 12, 2017	5:30-8:30 p.m.
27799	50	Thursday, December 14, 2017	8:30-11:30 a.m.
21111	50	Thursday, December 14, 2017	1:00-4:00 p.m.
21112	50	Monday, December 18, 2017	8:30-11:30 a.m.
21120	50	Monday, December 18, 2017	5:30-8:30 p.m.
21122	50	Friday, December 15, 2017	1:00-4:00 p.m.
21127	50	Friday, December 15, 2017	5:30-8:30 p.m.
21128	50	Sunday, December 17, 2017	8:30-11:30 a.m.
21228	50	Sunday, December 17, 2017	1:00-4:00 p.m.
21235	50	Sunday, December 17, 2017	5:30-8:30 p.m.
21240	50	Monday, December 18, 2017	8:30-11:30 a.m.
21241	50	Friday, December 15, 2017	5:30-8:30 p.m.
21242	50	Monday, December 18, 2017	1:00-4:00 p.m.
21256	50	Friday, December 15, 2017	8:30-11:30 a.m.
21257	50	Monday, December 11, 2017	1:00-4:00 p.m.
21259	50	Monday, December 11, 2017	5:30-8:30 p.m.
21260	50	Monday, December 18, 2017	5:30-8:30 p.m.
21268	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
21300	50	Tuesday, December 12, 2017	5:30-8:30 p.m.
21301	50	Thursday, December 14, 2017	8:30-11:30 a.m.
21325	50	Thursday, December 14, 2017	1:00-4:00 p.m.
21341	50	Thursday, December 14, 2017	5:30-8:30 p.m.
21355	50	Friday, December 15, 2017	8:30-11:30 a.m.
21356	50	Friday, December 15, 2017	1:00-4:00 p.m.
21369	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
21370	50	Sunday, December 17, 2017	8:30-11:30 a.m.
21371	50	Sunday, December 17, 2017	1:00-4:00 p.m.
21373	50	Tuesday, December 12, 2017	1:00-4:00 p.m.
21378	50	Monday, December 18, 2017	8:30-11:30 a.m.
21441	50	Monday, December 18, 2017	1:00-4:00 p.m.
21602	50	Monday, December 18, 2017	5:30-8:30 p.m.
21603	50	Monday, December 11, 2017	8:30-11:30 a.m.
21632	100	Monday, December 18, 2017	5:30-8:30 p.m.
21651	100	Tuesday, December 12, 2017	5:30-8:30 p.m.
21720	100	Monday, December 18, 2017	5:30-8:30 p.m.

24101	100	Friday, December 15, 2017	1:00-4:00 p.m.
24202	100	Monday, December 18, 2017	5:30-8:30 p.m.
24221	100	Monday, December 18, 2017	8:30-11:30 a.m.
24322	100	Sunday, December 17, 2017	1:00-4:00 p.m.
24334	100	Friday, December 15, 2017	5:30-8:30 p.m.
24351	100	Tuesday, December 12, 2017	8:30-11:30 a.m.
24370	100	Thursday, December 14, 2017	5:30-8:30 p.m.
24424	100	Thursday, December 14, 2017	1:00-4:00 p.m.
24425	100	Tuesday, December 12, 2017	8:30-11:30 a.m.
24451	100	Friday, December 15, 2017	1:00-4:00 p.m.
24626	100	Friday, December 15, 2017	8:30-11:30 a.m.
24652	100	Monday, December 18, 2017	8:30-11:30 a.m.
24683	100	Thursday, December 14, 2017	1:00-4:00 p.m.
24688	100	Friday, December 15, 2017	5:30-8:30 p.m.
24691	100	Friday, December 15, 2017	5:30-8:30 p.m.
24704	100	Tuesday, December 12, 2017	5:30-8:30 p.m.
24711	100	Monday, December 18, 2017	8:30-11:30 a.m.
24718	100	Sunday, December 17, 2017	1:00-4:00 p.m.
24722	100	Thursday, December 14, 2017	1:00-4:00 p.m.
24740	100	Monday, December 18, 2017	1:00-4:00 p.m.
24771	100	Tuesday, December 12, 2017	8:30-11:30 a.m.
24774	100	Thursday, December 14, 2017	8:30-11:30 a.m.
82101	100	Monday, December 11, 2017	8:30-11:30 a.m.
82102	100	Sunday, December 17, 2017	8:30-11:30 a.m.
82103	100	Sunday, December 17, 2017	5:30-8:30 p.m.
82104	100	Sunday, December 17, 2017	8:30-11:30 a.m.
82111	100	Friday, December 15, 2017	1:00-4:00 p.m.
82121	100	Monday, December 11, 2017	5:30-8:30 p.m.
82122	100	Monday, December 11, 2017	5:30-8:30 p.m.
82141	100	Monday, December 11, 2017	5:30-8:30 p.m.
82142	100	Tuesday, December 12, 2017	5:30-8:30 p.m.
82143	100	Monday, December 18, 2017	8:30-11:30 a.m.
82171	100	Tuesday, December 12, 2017	5:30-8:30 p.m.
82172	100	Thursday, December 14, 2017	8:30-11:30 a.m.
82173	100	Monday, December 11, 2017	5:30-8:30 p.m.
82174	100	Sunday, December 17, 2017	1:00-4:00 p.m.
82201	100	Tuesday, December 12, 2017	8:30-11:30 a.m.
82208	100	Sunday, December 17, 2017	5:30-8:30 p.m.
82211	100	Monday, December 18, 2017	1:00-4:00 p.m.

82241	100	Monday, December 11, 2017	1:00-4:00 p.m.
82242	100	Tuesday, December 12, 2017	1:00-4:00 p.m.
82271	100	Sunday, December 17, 2017	5:30-8:30 p.m.
82281	100	Monday, December 18, 2017	5:30-8:30 p.m.
82283	100	Tuesday, December 12, 2017	1:00-4:00 p.m.
82291	100	Tuesday, December 12, 2017	8:30-11:30 a.m.
82311	100	Monday, December 11, 2017	5:30-8:30 p.m.
82342	100	Monday, December 18, 2017	1:00-4:00 p.m.
82343	100	Thursday, December 14, 2017	5:30-8:30 p.m.
82345	100	Friday, December 15, 2017	1:00-4:00 p.m.
82373	100	Tuesday, December 12, 2017	5:30-8:30 p.m.
82411	100	Monday, December 18, 2017	8:30-11:30 a.m.
82425	100	Monday, December 18, 2017	1:00-4:00 p.m.
82444	100	Friday, December 15, 2017	5:30-8:30 p.m.
82455	100	Friday, December 15, 2017	8:30-11:30 a.m.
57149	100	Thursday, December 14, 2017	5:30-8:30 p.m.
57151	100	Tuesday, December 12, 2017	8:30-11:30 a.m.
57152	100	Tuesday, December 12, 2017	1:00-4:00 p.m.
57173	100	Thursday, December 14, 2017	5:30-8:30 p.m.
57284	100	Thursday, December 14, 2017	8:30-11:30 a.m.
57480	100	Tuesday, December 12, 2017	1:00-4:00 p.m.
57780	100	Tuesday, December 12, 2017	5:30-8:30 p.m.
80180	100	Thursday, December 14, 2017	1:00-4:00 p.m.
80211	100	Friday, December 15, 2017	5:30-8:30 p.m.
80212	100	Sunday, December 17, 2017	5:30-8:30 p.m.
80223	100	Sunday, December 17, 2017	5:30-8:30 p.m.
80282	100	Thursday, December 14, 2017	1:00-4:00 p.m.
80327	100	Monday, December 11, 2017	1:00-4:00 p.m.
80381	100	Friday, December 15, 2017	8:30-11:30 a.m.
80627	100	Friday, December 15, 2017	5:30-8:30 p.m.
80681	100	Thursday, December 14, 2017	5:30-8:30 p.m.
33115	100	Monday, December 18, 2017	5:30-8:30 p.m.
33121	100	Sunday, December 17, 2017	8:30-11:30 a.m.
33122	100	Thursday, December 14, 2017	5:30-8:30 p.m.
33124	100	Tuesday, December 12, 2017	1:00-4:00 p.m.
33141	100	Friday, December 15, 2017	8:30-11:30 a.m.
33142	100	Tuesday, December 12, 2017	5:30-8:30 p.m.
33151	100	Tuesday, December 12, 2017	1:00-4:00 p.m.
33211	100	Monday, December 11, 2017	8:30-11:30 a.m.

33231	100	Sunday, December 17, 2017	1:00-4:00 p.m.
33331	100	Thursday, December 14, 2017	1:00-4:00 p.m.
33338	100	Monday, December 11, 2017	5:30-8:30 p.m.
33341	100	Monday, December 18, 2017	8:30-11:30 a.m.
33441	100	Monday, December 18, 2017	5:30-8:30 p.m.
33445	100	Monday, December 11, 2017	8:30-11:30 a.m.
33650	100	Friday, December 15, 2017	8:30-11:30 a.m.
33755	100	Sunday, December 17, 2017	8:30-11:30 a.m.
33759	100	Sunday, December 17, 2017	8:30-11:30 a.m.
33761	100	Monday, December 11, 2017	1:00-4:00 p.m.
33778	100	Thursday, December 14, 2017	5:30-8:30 p.m.
33779	100	Monday, December 18, 2017	5:30-8:30 p.m.
33783	100	Sunday, December 17, 2017	1:00-4:00 p.m.
85102	100	Monday, December 18, 2017	1:00-4:00 p.m.
85211	100	Monday, December 18, 2017	8:30-11:30 a.m.
85213	100	Monday, December 11, 2017	1:00-4:00 p.m.
85219	100	Friday, December 15, 2017	8:30-11:30 a.m.
85241	100	Sunday, December 17, 2017	8:30-11:30 a.m.
85320	100	Tuesday, December 12, 2017	5:30-8:30 p.m.
85340	100	Tuesday, December 12, 2017	5:30-8:30 p.m.
85341	100	Sunday, December 17, 2017	8:30-11:30 a.m.
85370	100	Tuesday, December 12, 2017	8:30-11:30 a.m.
85390	100	Monday, December 18, 2017	1:00-4:00 p.m.
85408	100	Thursday, December 14, 2017	5:30-8:30 p.m.
85414	100	Monday, December 11, 2017	8:30-11:30 a.m.
85484	100	Tuesday, December 12, 2017	8:30-11:30 a.m.
85738	100	Monday, December 11, 2017	8:30-11:30 a.m.
85744	100	Monday, December 11, 2017	8:30-11:30 a.m.
85765	100	Thursday, December 14, 2017	8:30-11:30 a.m.
85770	100	Monday, December 11, 2017	1:00-4:00 p.m.
16161	100	Sunday, December 17, 2017	8:30-11:30 a.m.
16384	100	Thursday, December 14, 2017	8:30-11:30 a.m.
16456	100	Sunday, December 17, 2017	5:30-8:30 p.m.
16722	100	Thursday, December 14, 2017	5:30-8:30 p.m.
16811	100	Thursday, December 14, 2017	1:00-4:00 p.m.
16822	100	Tuesday, December 12, 2017	1:00-4:00 p.m.
88150	100	Friday, December 15, 2017	1:00-4:00 p.m.
88230	100	Tuesday, December 12, 2017	8:30-11:30 a.m.
88251	100	Friday, December 15, 2017	1:00-4:00 p.m.

88302	100	Thursday, December 14, 2017	1:00-4:00 p.m.
88341	100	Friday, December 15, 2017	1:00-4:00 p.m.
88411	100	Sunday, December 17, 2017	8:30-11:30 a.m.
17651	100	Sunday, December 17, 2017	1:00-4:00 p.m.
36200	100	Monday, December 11, 2017	5:30-8:30 p.m.
36201	100	Monday, December 18, 2017	1:00-4:00 p.m.
36202	100	Sunday, December 17, 2017	1:00-4:00 p.m.
36208	100	Friday, December 15, 2017	5:30-8:30 p.m.
36217	100	Tuesday, December 12, 2017	5:30-8:30 p.m.
36220	100	Sunday, December 17, 2017	8:30-11:30 a.m.
36225	100	Friday, December 15, 2017	8:30-11:30 a.m.
36309	100	Thursday, December 14, 2017	8:30-11:30 a.m.
36461	100	Thursday, December 14, 2017	1:00-4:00 p.m.
36661	100	Tuesday, December 12, 2017	8:30-11:30 a.m.
36700	100	Thursday, December 14, 2017	8:30-11:30 a.m.
36705	100	Thursday, December 14, 2017	5:30-8:30 p.m.
36707	100	Friday, December 15, 2017	5:30-8:30 p.m.
36749	100	Monday, December 18, 2017	5:30-8:30 p.m.