CSE 2010, HW2
Due Thu Feb 13 at the start of your lab section; Submit Server: class = cse2010, assignment = hw2SxIndividual
Due Thu Feb 13 at the end of your lab section; Submit Server: class = cse2010, assignment = hw2SxGroupHelp
x is 14, 23 or j—(j for java).

1 Written Part (30 points)

1. Explain the number of additions to the total (not Big-O) in terms of n for the following program segment:

```
int total = 0;
for (int i = n; i > 0; i -= 2)
  total += i;   // addition to the total
```

2. Explain the number of additions to the total (not Big-O) in terms of n for the following program segment:

```
int total = 0;
for (int i = 0; i < n; i++)
  for (int j = i; j >= 0; j--)
    total += i * j; // addition to the total
```

3. Mathematically show that if $d(n)$ is $O(f(n))$ and $e(n)$ is $O(g(n))$, $d(n) + e(n)$ is $O(f(n) + g(n))$.

4. Consider $f(n) = 4n^2 + 3n - 2$, mathematically show that $f(n)$ is $O(n^2)$, $\Omega(n^2)$, and $\Theta(n^2)$.

5. For finding an item in a sorted array, consider "ternary search" (similar to binary search). It compares array elements at two locations and eliminates 2/3 of the array. To analyze the number of comparisons, the recurrence equations are $T(n) = 2 + T(n/3)$, $T(2) = 2$, and $T(1) = 1$, where n is the size of the array. Explain why the equations characterize "ternary search" and solve for $T(n)$.

6. To analyze the time complexity of the "brute-force" algorithm in the programming part of this assignment, we would like to count the number of all possible schedules.
   (a) Explain the number of all possible schedules in terms of n (number of candidate courses) and m (number of time slots per candidate course) in the worst case.
   (b) Consider a computer that can process 1 billion schedules per second, n is 10, m is 20, explain the number of hours needed to process all possible schedules.
   (c) If we do not want the computer to spend more than 1 minute, explain the largest n the computer can process when m is 20.

2 Programming Part (70 points)

Many businesses need to schedule workers for shifts. The worker scheduling problem (which includes times of shifts, number of employees per shift, workers' availability and so on) could be quite complex. We are going to investigate a similar but simpler scheduling problem that you are familiar with. During course registration each semester, students need to select courses that do not have a time conflict with each other. Students usually have a set of candidate courses and each course might have multiple sections (at different times).

The goal of this assignment is to automate part of the student course scheduling process. That is, you can save time in course scheduling in the future (Yeah!). Given a set of candidate courses and their time slots (sections), we would like the *largest* subset of courses that do not have a time conflict. Also, a student can specify the preference order of candidate courses and time slots (sections) for each course. If two courses are in the same time slot, the course with a higher preference is considered first. If a course has multiple time slots, the time slot with a higher preference is considered earlier. That is, we would like to find the "best" schedule that has the largest number of courses (first priority) with the most preferred courses (second priority) and time slots (third priority).

Design a "brute-force" algorithm that **recursively** enumerates all possible schedules and find the "best" schedule. For simplicity, all courses are in standard time slots—MWF on the hour or TR every 1.5 hours starting at 0800 (and ending before 1700), MW or TR every 1.5 hours starting at 1700 (and ending before 2130). That is, the time slots do not overlap. Suggestion/hint: consider the correctness of your algorithm before its efficiency. Sample input and sample output are on the course website.

We will evaluate your submissions on code01.fit.edu so we strongly recommend you to test your programs on code01.fit.edu. To preserve invisible characters, we strongly recommend you to download, NOT copy and paste, input data files.

Input (from a file): The command-line argument for hw2.c is the name of the input file. Each line has the course number and its time slots (sections). The candidate courses are ordered according to the student's preference. If a course has multiple time slots (sections), they are ordered according to the student's preference. For example:

```
CSE4301 MWF0900
MTH2001 MWF0900 MWF1500
PHY1001 MWF1500
COM2223 MWF1100 TR0930 TR1530 MWF1000
```

Output (to the screen): Course schedule with the course number and its time slot on each line (same ordering as input, ie, student's preference). The course schedule is followed by courses with a time conflict and their time slots (same ordering as input, ie, student's preference) or "None". For example,

```
---Course schedule---
CSE4301 MWF0900
MTH2001 MWF1500
COM2223 MWF1100
---Courses with a time conflict---
PHY1001 MWF1500
```

Extra Credit (10 points): Separate submission via hw2extra.c. Solve the problem without recursion (or using a stack to simulate recursion).

3 Submission

Submit hw2.c that has the main method and other program files. Submissions for Individual and GroupHelp have the same guidelines as HW1.

Submit the written part in PDF format to the Submit Server. Hardcopy is also acceptable in the lab. GroupHelp submission is not applicable to the written part.

Note the late penalty on the syllabus if you submit after the due date and time as specified at the top.

For extra credit, submit hw2Extra.c that has the main method and other program files. GroupHelp submission is not applicable to extra credit. Late submission for extra credit is not accepted.