CSE 3353 Homework 1 Fall 2014

Assignment is due at 11am on September 11. Submit a scanned copy of the assignment, including a copy of your code and outputs as requested in the assignment, on Scryst. You should turn in a copy of the code in a .zip file. The Python file must be named job_selection.py, and you must not modify the names of any of the functions or their defined parameters. Any outputs from running the code should be included in a file called job_selection_output.txt, also included in the zip file. Both of these files must be included in a directory called LastnameFirstname. Please write your name(s) as a comment in the first line of code in job_selection.py. If you are working with a partner, turn in one set of code and one set of answers for questions Q2–Q11.

You may submit a lateness coupon request BEFORE the assignment is due by sending a private Piazza message with Subject “CSE3353 Lateness Coupon”. All other late work will receive a 10 percentage point deduction per day (including weekends). No late work is accepted beyond one week after the assignment is due.

Q1 (50) ________
Q2 (3) ________
Q3 (3) ________
Q4 (6) ________
Q5 (12) ________
Q6 (5) ________
Q7 (6) ________
Q8 (5) ________
Q9 (3) ________
Q10 (6) ________
Q11 (1) ________
Total (100) ________
Q1. (50 points)
Your task is to implement in Python and evaluate the performance of multiple algorithms solving the "Movie-Scheduling Problem" described in Lecture 4. Starter code is available at [http://lyle.smu.edu/~tylerm/courses/cse3353/code/job_selection_starter.txt](http://lyle.smu.edu/~tylerm/courses/cse3353/code/job_selection_starter.txt) Some functions have been implemented for you, as they require Python-specific knowledge not covered in depth in the lecture.

Movies are represented as a dictionary with the names of the movies as keys mapping to a 2-tuple of datetime objects representing the start and end times for the movie:

```python
movieTimes = {
    "Tarzan of the Jungle":
    (datetime.date(2013,3,1),datetime.date(2013,10,15)),
}
```

The testing code in parts (b,e,f) should be implemented inside the `if __name__=='__main__':` clause in the main body of the code.

a. Each of the job-selection algorithms must check for overlapping movie times. To promote good abstraction and modularity practice, you should first write a function `checkOverlap` that compares two time intervals and returns True if there is an overlap and False otherwise.

b. Write test code to verify that overlapping times and non-overlapping times are correctly identified. You should try multiple cases where overlap does, or does not, exist to confirm that the correct assessment has been made.

c. Implement the `exhaustiveScheduling` function based on the following pseudo-code:

```python
ExhaustiveScheduling(I)
    j = 0
    Smax = {}
    For each of the 2^n subsets Si of intervals I
        If (Si is mutually non-overlapping) and (size(Si)>j)
            then j = size(Si) and Smax = Si
    Return Smax
```

Write code to check all subsets of movies to find the most movies that can be selected without overlaps. *(Note that you should make use of the subset() function provided in the sample code.)*

d. Write code to implement the `OptimalScheduling` algorithm based on the following pseudo-code:

```python
earliestFinish(I)
    movie_titles = list of movie titles in I sorted by earliest completion date.
    job_list = empty list
    While (movie_titles is non-empty) do
        Add job j with the earliest completion date to job_list
        Delete j, and whatever intersects with j, from movie_titles.
    return job_list
```

The code should be placed in the function `earliestFinish`. *(Hint: you should review how the provided code implements the incorrect earliestJobFirst algorithm to help guide your implementation of earliestFinish.)*

e. Having implemented two algorithms in parts (c) and (d), you should now run tests to become confident in their correctness. First, create a movie-time dictionary demonstrating that the provided `earliestJobFirst` function returns a suboptimal movie schedule but the `earliestFinish` function provides the optimal solution.

f. As a second test of the algorithms written in parts (c) and (d), write test code showing that `exhaustiveScheduling` and `earliestFinish` give the same answer (number of movies), and that both appear correct, for 15 movies generated using the `generateJobs` function. Print the movie names, plus start and end times, for movies selected by each algorithm. Also, print the movie names, plus start and end times, for movies not selected by the algorithm. Be sure to print the movies in chronological order.

g. Implement the function `compareSpeed` to compare execution times for `earliestFinish` and `exhaustiveScheduling` for 10, 15, and 20 movies created by the `generateJobs` function. Use Python's `timeit` library and execute the function 100 times for `earliestFinish` and 5 times for `exhaustiveScheduling`. Print the times for each algorithm and number of movies. Include the printout with your assignment. *(Note: if you have to wait more than 20 minutes for the exhaustive code to complete, you can kill the task, indicate where the code got stuck, and include only the output that completed.)*
Q2. (3 points) Decide whether the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

In every instance of the Stable Matching Problem, there is a stable matching containing a pair $(m, w)$ such that $m$ is ranked first on the preference list of $w$ and $w$ is ranked first on the preference list of $m$.

Q3. (3 points) Decide whether the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

Consider an instance of the Stable Matching Problem in which there exists a man $m$ and woman $w$ such that $m$ is ranked first on the preference list of $w$ and $w$ is ranked first on the preference list of $m$. Then in every stable matching $S$ for this instance, the pair $(m, w)$ belongs to $S$. 
Q4. (6 points)
State whether the following is True or False, along with a brief explanation of your reasoning.

a. \(2^n + 2 = O(2^n)\)?

b. \(n \cdot 2^n = O(2^n)\)?
Q5. (12 points)
For each of the following pairs of functions, either \( f(n) \) is in \( O(g(n)) \), \( f(n) \) is in \( \Omega(g(n)) \), or \( f(n) = \Theta(g(n)) \) (i.e., \( f(n) \) is in both \( O(g(n)) \) and \( \Omega(g(n)) \)). Determine which relationship is correct and briefly explain why.

a. \( f(n) = \log_2(2n); g(n) = \log_2(n) + 2 \)

b. \( f(n) = \sqrt{n}; g(n) = \log(n^2) \)

c. \( f(n) = \log 100; g(n) = 5 \)

d. \( f(n) = \log n + n; g(n) = n \log n \)

e. \( f(n) = 2^n; g(n) = n^2 \)

f. \( f(n) = 3^n; g(n) = 2^n \)
Q6. (5 points) True or False?

a. \(2n^2 + 1 = \Omega(n^2)\)

b. \(\log n = o(\sqrt{n})\)

c. \(\sqrt{n} = o(\log n)\)

d. \(\sqrt{n} \log(n) = O(n \log n)\)

e. \(n \log(n) = O(\sqrt{n} \log n)\)
Q7. (6 points)
For each of the following pairs of functions \( f(n) \) and \( g(n) \), state whether \( f(n) = O(g(n)) \), \( f(n) = \Omega(g(n)) \), \( f(n) = \Theta(g(n)) \) (i.e., \( f(n) \) is in both \( O(g(n)) \) and \( \Omega(g(n)) \)), or none of the above.

a. \( f(n) = n^2 + 3n + 4 \), \( g(n) = 6n + 7 \)

b. \( f(n) = n\sqrt{n} \), \( g(n) = n^2 - n \)

c. \( f(n) = 2^n - n^2 \), \( g(n) = n^4 + n^2 \)
Q8. (5 points)
For each of these questions, briefly explain your answer.

a. Explain why saying “The running time for my algorithm is at least $O(n^2)$” is meaningless.

b. If I prove that an algorithm takes $O(n^2)$ worst-case time, is it possible that it takes $O(n^3)$ on some inputs?

c. If I prove that an algorithm takes $O(n^2)$ worst-case time, is it possible that it takes $O(n)$ on all inputs?

d. If I prove that an algorithm takes $\Theta(n \log n)$ worst-case time, is it possible that it takes $O(n)$ on some inputs?

e. If I prove that an algorithm takes $\Theta(n \log n)$ worst-case time, is it possible that it takes $O(n)$ on all inputs?
Q9. (3 points)
Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.

\[ f_1(n) = n^{2.5} \]
\[ f_2(n) = \sqrt{2n} \]
\[ f_3(n) = n + 10 \]
\[ f_4(n) = 10^n \]
\[ f_5(n) = 100^n \]
\[ f_6(n) = n^2 \log n \]
Q10. (6 points)
Assume you have functions \( f \) and \( g \) such that \( f(n) \) is \( O(g(n)) \). Decide whether each of the following statements is true or false and give a proof or counterexample.

a. \( 2f(n) \) is \( O(2g(n)) \).

b. \( f(n)^2 \) is \( O(g(n)^2) \).

Q11. (1 point) How long (in hours) did you spend on this assignment? Please estimate separately how long you spent on programming (Q1) and the other questions (Q2–Q10) (full credit for any truthful answer)