CSCI 3753: Operating Systems
Fall 2016
Extra Credit Problem Set

- Please write your answers in the space provided.
- You may earn up to 25 points extra credit for your midterm grade from this problem set.
- **Due date:** Thursday, November 17 in class. No extensions will be given except at the instructor's discretion in documented cases of extreme hardship or emergencies. Please submit a hardcopy of your solutions.
- **Absolutely no collaboration with anyone is allowed for this exam. Please refrain from discussing your questions/answers with your classmates or anyone else. Failure to adhere to this policy will result in a grade of zero.**

Problem 1. [10 Points] The IA-32 architecture provides an atomic CAS (Compare and Swap) instruction. CAS mimics the following code:

```
int CAS (int *mem , int testval , int newval)
{
   int res = *mem;
   if (* mem == testval) *mem = newval;
   return res;
}
```

Provide a solution for mutual exclusion using *CAS* instruction. You may not use any other synchronization primitive.

Problem 2. [30 Points] Search/Insert/Delete. Three kinds of processes share access to a singly-linked list; Searchers merely examine the linked list; hence they can execute concurrently with each other. Inserters add new items to the end of the list; insertions must be mutually exclusive to preclude two inserters from inserting new items at the same time. However, one insert can proceed in parallel with any number of searches. Finally, deleters remove items from anywhere in the list. At most one deleter can access the list at a time, and deletion must also be mutually exclusive with searches and insertions. Give a solution to this problem. Use semaphores to implement the required mutual exclusion. Assume that you have procedures insert() to do insertions, delete() to do deletions, and search() to do searches.

Problem 3. [30 Points] River Crossing. A particular river crossing is shared by both Linux hackers and Microsoft employees. A single boat is used to cross the river, but it only seats **four** people, and must always carry a full load. In order to guarantee the safety of the hackers, you cannot put three employees and one hacker in the same boat; similarly, you cannot put three hackers in the same boat as an employee. All other combinations are safe. On arriving at the river bank, a hacker thread calls the function HackerArrives() and an employee thread calls the function EmployeeArrives(). The procedures arrange the arriving hackers and employees into safe boatloads. To get into the boat, a thread calls BoardBoat(); once the boat is full, *one* thread calls RowBoat(). RowBoat() does not return until the boat has dropped off the passengers on the other side and has returned empty to carry the next load.

Assume BoardBoat() and RowBoat() are already written. Provide an implementation os HackerArrives() and EmployeeArrives(). These methods should not return until after RowBoat() has been called for the boatload. Use monitor for synchronization. Your solution should be starvation free, there should be no busy-waiting and no undue waiting (hackers and employees should not wait if there are enough of them for a safe boatload).

Problem 4. [30 Points] Using semaphores, provide codes for processes X1, X2, X3, X4 and X5 in the following synchronization problem. Each process has a special (synchronization) point in its code.

- Process X1 may cross its synchronization point unconditionally,
- Process X4 has to wait for processes X3 and X1 to cross or reach their synchronization points, and only then X4 may cross its synchronization point,
- Process X2 has to wait for process X1 to cross or reach its synchronization point, and only then X2 may cross its synchronization point,
- Process X3 has to wait for process X4 to cross or reach its synchronization point, and only then X3 may cross its synchronization point,
- Process X5 has to wait for processes X1, X3 and X4 to cross their synchronization points, and only then X5 may cross its synchronization point.

Assume that each process will cross its synchronization point only once.