NAME_____

Write your name above.

Grading Rubric

There are four questions in this assignment, each worth the percentage indicated. Questions are assessed on their completeness and correctness and graded in accordance with the ***General Requirements for Assignments*** document posted on the course website.

Instructions for Submitting the Homework

1. **Due Date**: ***This assignment is due in class on November 14. You will not be submitting this assignment electronically.***
2. Your homework must comply with the ***General Requirements for Assignments*** document posted on the course website for full credit. Submissions that do not follow these instructions will not be graded and will be given a **zero grade**.
3. ***Print this assignment single-sided and fill in your answers in the space provided.***
4. Bring the completed assignment to class. It will not be accepted after the end of the class on November 14.

1 Questions

1. (20%) Suppose that three different processes, P_1, P_2, and P_3, share the variables `x` and `y` and execute the following code fragments concurrently, and that the initial value of `y` is 5. The instructions to add and subtract in memory are not atomic.

P_1	P_2	P_3
`x = y;`	`x = y;`	`x = y;`
`x = x + 1;`	`x = x + 1;`	`x = x - 1;`
`y = x;`	`y = x;`	`y = x;`

   (a) (10%) In the box below, write all possible final values of y.

   (b) (10%) If all of the individual instructions executed by these processes were atomic, how many different interleaved sequences could there be?

2. (30%) Peterson's algorithm is a solution to the two-process critical section problem. Below is an alternative algorithm that would be executed by each of two processes P_0 and P_1.

```
/* Global shared variables */
boolean flag[2] = {false, false};
int turn;  /* initially either 0 or 1, randomnly */

/* Process i executes the following loop, for i = 0, 1 */
while (true) {
    flag[i] = true;
    while ( flag[1-i] ) {
        if ( turn == 1-i ) {
            flag[i] = false;
            while ( turn == 1-i )
                ; /* do nothing */
            flag[i] = true;
        }
    }

    /* critical section */

    turn = 1-i;
    flag[i] = false;

    /* remainder section */
}
```

Which of the criteria for the critical section problem are satisfied? Put a check next to the box if it is satisfied.

☐ Mutual Exclusion

☐ Progress

☐ Bounded Waiting

3. Consider the following snapshot of a resource allocation system. There are no current outstanding, unsatisfied requests.

	Maximum Claim				**Current Allocation**				**Current Need**			
process	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	0	0	1	2	0	0	1	2				
P_2	2	7	5	0	2	0	0	0				
P_3	6	6	5	6	0	0	3	4				
P_4	4	3	5	6	2	3	5	4				
P_5	0	6	5	2	0	3	3	2				

Available			
R_1	R_2	R_3	R_4
2	1	0	0

(a) (10%) Compute what each process might still request and display it in the columns labeled "Current Need."

(b) (10%) Assuming that each process makes the requests you computed in the first question, is the resulting system in a safe or an unsafe state? If it is safe, provide a safe sequence.

(c) (5%) Is the resulting state a deadlock state?

(d) (5%) Which processes, if any, are or may become deadlocked?

4. (20%) A computer system has a total of 150 units of memory, currently allocated to three processes as shown below:

Process ID	Maximum Claim	Allocation
1	70	45
2	60	40
3	60	15

Apply the Banker's Algorithm to determine whether it would be safe to grant each of the following requests. If your answer is yes, provide a safe sequence that justifies the answer. If your answer is no, give a sequence that justifies this.

(a) A fourth process arrives with a maximum claim of 60 memory units and an initial request for 25 units.

(b) A fourth process arrives with a maximum claim of 60 memory units and an initial request for 35 units.