	Current Allocation				Maximum Demand				Available			
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	2	1	0	0
p_2	2	0	0	0	2	7	5	0				
p_3	0	0	3	4	6	6	5	6				
p_4	2	3	5	4	4	3	5	6				
p_5	0	3	3	2	0	6	5	2				

(a) [5 pt] Draw a process resource graph to show the above snapshot.

(b) [4 pt] Is the system currently in a safe or unsafe state? Why?

(c) [2 pt] Is the system currently deadlocked? Why or why not?

(d) [2 pt] Which processes, if any, are or may become deadlocked?

(e) [5 pt] If a request from p_3 arrives for $(0, 1, 0, 0)$, can that request be safely granted immediately? In what state (deadlocked, safe, unsafe) would immediately granting that request leave the system? Which processes, if any, are or may become deadlocked if this whole request is granted immediately?

4. [10 pt] Consider the following set of processes:

Process	Arrival Time	Processing Time
p_1	0	3
p_2	1	5
p_3	3	2
p_4	9	5
p_5	12	5

Compute the average waiting time and the average turnaround time for each of the following scheduling algorithms:

(a) FIFO

(b) LIFO

(c) SJF (same as SJN)

(d) Shortest remaining time next

(e) Round robin (quantum = 2)

CS 376 **Operating Systems** Test 2
Name: Maximum Points: 47

Important: This is an open book test. You can use any books, notes, or paper. *Do not log into the computer during the test.* Any calculations and rough work can be done on the back side of the test pages.

1. Consider the following program in the parallel C syntax:

```
#define n 50
int tally;
main()
{
   tally = 0;
   parbegin
      total();
      total();
   parend;
   printf ( "%d\n", tally );
}

void total();
{
   int count;
   for ( count = 1; count < n; count++ )
      tally++;
}
```

   (a) [8 pt] Determine the proper lower bound and upper bound on the final value of the shared variable `tally` output by this concurrent program. Assume processes can execute at any relative speed and that a value can be incremented only after it has been loaded into a register by a separate machine instruction.

   (b) [5 pt] Suppose that an arbitrary number of these processes are permitted to execute in parallel under the assumptions of part (a). What effect will this modification have on the range of final values of `tally`?

2. [6 pt] Show that the bakery algorithm avoids deadlock while enforcing mutual exclusion.

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