1. (21%) Consider the following snapshot of a system:

	Allocation	Matrix	Available
	ABCD	ABCD	ABCD
P0	0 0 1 2	0 0 1 2	1 5 2 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

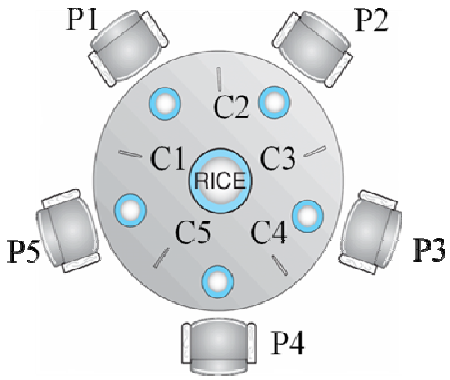
Answer the following questions using the banker's algorithm. Note that all the following questions are independent.

(a) Is the system in a safe state? Be sure to justify your answer.

(b) If a request from process P1 arrives for (0, 4, 2, 0), can the request be granted immediately? Be sure to justify your answer.

(c) If a request from process P4 arrives for (0, 4, 2, 0), can the request be granted immediately? Be sure to justify your answer.

2. (20%) For the dining philosophers problem, assume it has reached the deadlock state that each philosopher, P*i*, has acquired the chopstick on his right-hand side, C*i*, where 1 ≦ *i* ≦5. Please draw (a) the resource allocation graph and (b) the wait-for graph for the philosopher deadlock situation.



3. (9%) What are the advantages and disadvantages of using circuit switching?

4. (10%) What are two differences between user-level threads and kernel-level threads? Under what circumstances is one type better than the other?

5. (15%)Explain the purpose of the checkpoint mechanism. 5%

How often should checkpoints be performed? Describe how the frequency of checkpoints affects: 10%

- System performance when no failure occurs
- The time it takes to recover from a system crash
- The time it takes to recover from a disk crash

6. (25%)The first known correct software solution to the critical-section problem for n processes with a lower bound on waiting of n-1 turns was presented by Eisenberg and McGuire. The Processes share the following variables:

```
1.      enum pstate { idle, want_in, in_cs};
2.      pstate flag[n];
3.      int    turn;
4.
```

5. All the elements of flag are initially idle; the initial value of turn is immaterial(between 0 and n-1). The structure of process Pi is shown in the following:

```
6.  do {
7.  while(TRUE) {
8.              flag[i] = want_in;
9.              j = turn;
10.                     while (j != i) {
11.                        if (flag[j]    !=    idle) {
12.                           j = turn;
13.                        else
14.                           j = (j+1) % n;
15.                                                 }
16.              flag[i] = in_cs;
17.              j = 0;
18.              while ((j<n) && (j == i    ||    flag[j] != in_cs)) j++;
19.              if ((j >= n)    &&    (turn    == i    || flag[turn] == idle)) break;
20.             }
21.
22.         turn = i;
23.
24.         // CRICIAL-SECTION
25.
26.         j = (turn+1) % n;
27.         while (flag[j]    == idle)
28.            j = (j+1) %n;
29.         turn    = j;
30.         flag[i] = idle;
31.
32.         // REMAINDER-SECTION
33. } while(TRUE);
34.
```

(1).Prove that the algorithm satisfies all three requirements for the critical section. 15%
(2).What is the basic assumption for hardware requirements? If the system is multicore or multiprocessor system, the processor use the cache to speed up the reference of shared memory , what problems should be considered ? 10%