Name _____ Student No. _____

No aids allowed. Answer all questions on test paper. Use backs of sheets if necessary.

Total Marks: **40**

[10] 1. The article *Peer-to-Peer Systems* by Rodrigues & Druschel defines a P2P system as a distributed system with three properties; what are these properties? Explain them briefly.

Solution: High degree of decentralization: The peers implement both client and server functionality and most of the system's state and tasks are dynamically allocated among the peers. There are few if any dedicated nodes with centralized state. As a result, the bulk of the computation, bandwidth, and storage needed to operate the system are contributed by participating nodes. **Self-Organization:** Once a node is introduced into the system (typically by providing it with the IP address of a participating node and any necessary key material), little or no manual configuration is needed to maintain the system. **Multiple administrative domains:** The participating nodes are not owned and controlled by a single organization. In general, each node is owned and operated by an independent individual who voluntarily joins the system.

[10] 2. In the BitTorrent system, Alice decides which requests to respond to as follows: every 10 seconds, Alice recalculates the top 4 neighbors that supply her data at the highest rate.

But also, every 30 seconds Alice picks a neighbor at random, and sends it chunks. Describe the benefit of picking this random neighbor; what effect has it on the system as a whole?

Solution: Since Alice sends chunks to a new random neighbor — call him Bob — Alice may become Bob's "top 4 uploader." Thus, Bob will in turn start sending chunks to Alice, and become Alice's "top 4 uploader." The effect of this on the system is that peers able to upload at compatible rates will tend to find each other out.

[10] 3. Consider the following solution for the two-process synchronization, where below is the code for P_i (recall that, as usual, P_j is the *other* process where $i = 1 - j$).

```
do {
   flag[i] = true
   turn = j
   while (flag[j] and turn=j)

   CRITICAL SECTION

   flag[i] = false

   REMAINDER SECTION

} while 1
```

Show that the following three hold:

(a) Mutual exclusion is preserved

(b) Progress requirement is satisfied

(c) Bounded waiting requirement is met

(Use next blank page if needed.)

Solution: Mutex: suppose P_0, P_1 are both in the critical section. That would mean that `flag[0]=flag[1]=true` and so they both entered their respective critical sections by breaking out of the while-loop with `turn=0` (for P_0) and `turn=1` (for P_1). But `turn` is a shared variable, and hence this is not possible.

Progress: suppose that P_0 is ready for another round of CS, and that P_1 is in its RS. That means that P_1 just set `flag[1]=false` which breaks P_0 out of the while loop. The dual argument shows the same for P_1 ready for CS and P_0 in its RS.

Bounded waiting: If P_0 is ready to enter its CS, then P_1 will be able to enter its CS at most once; as soon as P_1 exits its CS, it turns `flag[1]=false` allowing P_0 entry. Again, the dual argument shows the same for P_1 and P_0.

Blank page.

[10] 4. Consider the *Bakery* multi-process synchronization solution:

The shared variables are: `choosing[i]`$\in \{0,1\}$, for all i, initially all 0, writable by i and readable by all $j \neq i$. Also, `number[i]`$\in \mathbb{N}$, initially all 0, writeable by i and readable by all $j \neq i$.

Process P_i:

```
0.  do {
1.    choosing[i]=1
2.    number[i]=1+max{j != i: number[j]}
3.    choosing[i]=0
4.    for j != i:
5.      waitfor choosing[j]=0
6.      waitfor number[j]=0 or (number[i],i)<(number[j],j)
7.    CRITICAL SECTION
8.    number[i]=0
9.    REMAINDER SECTION
10. } while 1
```

Recall that $(a,b) < (c,d)$ iff $a < c$ or $(a = c$ and $b < d)$.

(a) We call the three lines of code from line 1. until line 3. the *doorway*. Can more than one process be in the doorway at the same time?

(b) Suppose $i \neq j$ and P_i is in CS (line 7.) while P_j is somewhere in the range 4–7 (4 and 7 included). Then, show that necessarily `(number[i],i)<(number[j],j)`.

(c) Use part (b) to show Mutual Exclusion.

Solution: (a) Yes, they can; and thus it is possible for two processes to get the same ticket value (i.e., `number`). Those ties are broken in line 6.

(b) Let t_1 be some point in the execution where P_i is in CS (line 7.) while P_j is somewhere in the range 4–7 (4 and 7 included). Let t_2 be the point at which P_i reads `choosing[j]=0` in line 5. and thus moves on from line 5. We know that $t_2 < t_1$ (i.e., t_2 happens earlier than t_1).

Since P_j is in 4–7, it means that P_j was at time t_3 in the "choosing region" of the doorway, i.e., on line 2., where $t_3 < t_1$. We now compare t_2 and t_3.

If $t_2 < t_3$ then that means that P_i executed line 5. *before* P_j was in the doorway. Hence, when P_j was choosing, `number[i]` was already set, and so by line 2. `number[j]>number[i]` which implies `(number[i],i)<(number[j],j)`.

If $t_2 > t_3$ then P_j leaves the doorway before P_i executes line 5.; thus, when P_i is in line 6., P_j already chose its number. But since P_i entered its CS anyway, it must be the case that `(number[i],i)<(number[j],j)`.

Blank page.