**Name__________________________________________________**

**CSC552 Final, Take-home Spring 2014**

[40 points]

**I. Short Answer:** Concisely respond to each of the questions given. Answers should answer the question asked in detail but correspond to number of points of the question.

1. We discussed 6 possible solutions to achieve mutual exclusion. Briefly explain 4 of these. Include a brief description as well as any advantages or disadvantages. [12 points – 3 each]

2. Describe how message passing (send() and recv()) can be used to provide mutual exclusion. Include in your explain what send() and recv() do and how they would be used. [8 points]

3. Explain the implementation differences in the last two programs (shared memory vs. message queues), including such things as complexity, efficiency, correctness, etc. Also include a comparison of run-times between the two implementations. [10 points]

4. Answer just ONE of a or b for this question.   [10 points]

   a. This is a possible solution for the producer-consumer problem.

```
#define  N    100                  /* number of slots in the buffer */
int count = 0;                     /* number of items in the buffer  */

void producer()
{
    while (TRUE)  {                /* loop forever */
        produce_item();            /* generate next item */
        if (count == N) sleep();   /* if buffer is full, go to sleep */
        enter_item();              /* put item in buffer */
        count = count + 1;         /* increment count of items in buffer */
        if (count == 1) wakeup(consumer);   /* was buffer empty? */
    }
}   /* end producer */

void consumer()
{
    while (TRUE)  {                /* loop forever */
        if (count == 0) sleep (); /* if buffer is empty, go to sleep */
        remove_item();             /* take item out of buffer */
        count = count – 1;/* decrement count of items in buffer */
        if (count == N-1) wakeup(producer);    /* was buffer full? */
        consume_item();  /* print item */
    }
}   /* end consumer */
```

   What is wrong with this code (be specific and explain your answer)? How can it be corrected?

b. This is another possible solution to the producer-consumer problem.

```
#define  N    100                  /* number of slots in the buffer */
typedef  int  semaphore;           /* semaphores are a special kind of int */
semaphore  mutex = 1;   /* controls access to critical section */
semaphore empty = N;    /* counts empty buffer slots */
semaphore full = 0;                /* counts full buffer slots */

void producer()
{
      int item;
    while (TRUE)  {                /* loop forever */
        produce_item(&item);       /* generate next item */
        down(&mutex);    /* enter critical section */
        down(&empty);              /* decrement empty count */
        enter_item();              /* put item in buffer */
        up(&mutex);                /* leave critical section */
        up(&full);                 /* increment count of full slots */
    }
}   /* end producer */

void consumer()
{
    int item;

    while (TRUE)  {                /* loop forever */
        down(&full);               /* decrement full count */
        down(&mutex);              /* enter critical section */
        remove_item(&item);        /* take item out of buffer */
        up(&mutex);                /* leave critical section */
        up(&empty);                /* increment count of empty slots */
        consume_item(item);        /* do something with item */
    }
}   /* end consumer */
```

What is wrong with this code (be specific and explain your answer)? How can it be corrected?