1. Answer in one sentence to each of the following questions: (6×5 points)
   (a) What is the major advantage of user-level threads?
   (b) What is the difference between virtual circuits and datagrams?
   (c) What is the major limitation of non-preemptive scheduling policies?
   (d) Why is the fork() system call so expensive?
   (e) What is the main reason to use busy waits in multiprocessor architectures?
   (f) What is the main disadvantage of atomic translations?

2. Consider the following System V Release 4 scheduler:

<table>
<thead>
<tr>
<th>#ts_quantum</th>
<th>ts_tqexp</th>
<th>ts_slpret</th>
<th>ts_maxwait</th>
<th>ts_lwait</th>
<th>LEVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0</td>
<td>1</td>
<td>8000</td>
<td>1</td>
<td># 0</td>
</tr>
<tr>
<td>500</td>
<td>0</td>
<td>2</td>
<td>4000</td>
<td>2</td>
<td># 1</td>
</tr>
<tr>
<td>200</td>
<td>1</td>
<td>3</td>
<td>1600</td>
<td>3</td>
<td># 2</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>3</td>
<td>800</td>
<td>3</td>
<td># 3</td>
</tr>
</tbody>
</table>

   (a) What is the lowest priority level a process can have? (5 points) Answer: _0_
   (b) If a process at priority level 2 gets CPU for 100 ms and does a system call, what will be its new priority level when it returns to the ready queue? (5 points) Answer: _3_

   (c) Does the scheduler protect all processes from the risk of starvation? (5 points) Justify your answer. (5 points)

3. What is wrong with the following solution to the mutual exclusion problem? (10 points to tell me when and how it fails)

   //Global variables
   int tie_breaker;
   int want_in[2] = {0, 0};

   void enter_region(int pid) {
       tie_breaker = pid;
       want_in[pid] = 1;
       while (want_in[1 - pid] && tie_breaker != pid);
   } // enter_region

   void leave_region(int pid) {
       want_in[pid] = 0;
   } // leave_region

   They do not provide mutual exclusion under almost any circumstance!
4. An interstate bus that can carry 40 passengers, has a single door that let one passenger get in or out at any time. Add semaphores to the following two functions to ensure that the bus will never be overloaded and (b) passengers will not collide with each other when embarking or debarking the bus. (20 points minus 2 points if you forget to initialize the semaphores)

```c
semaphore passengers = __40__;
semaphore enter = __1__;

void debark()
{
    
    P(&enter);
    get_out();
    V(&enter);
    V(&passengers);
}

void embark()
{
    P(&passengers);
    P(&enter);
    get_in();
    V(&enter);
    V(&passengers);
}
```

5. What is the Unix system call that processes use to request a connection to another process? (5 easy points)

6. How does the VMS scheduler handle processes that return to the ready state after completion of their I/O request? (15 easy points)
This exam is closed book. You can have one page of notes. People caught cheating will be expelled from UH.

1. Answer in one sentence to each of the following questions: (6×5 points)
   (a) What is the major disadvantage of user-level threads?
   (b) What is the difference between virtual circuits and streams?
   (c) What is the major limitation of UNIX pipes?
   (d) What is the main disadvantage of busy waits in uniprocessor architectures?
   (e) What is the main advantage of atomic translations?
   (f) Why is the fork() system call so expensive?

2. Consider the following System V Release 4 scheduler:

   #ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait LEVEL
   1000 0 1 8000 0 # 0
   500 0 2 4000 2 # 1
   200 1 3 1600 3 # 2
   100 2 3 800 3 # 3

   (a) What is the highest priority level a process can have? (5 points)
   Answer: _____

   (b) If a process at priority level 1 waits for 1200 ms in the ready queue and has the CPU for 200 ms before returning to the ready queue, what will be its new priority? (5 points)
   Answer: _____

   (c) Does the scheduler protect all processes from the risk of starvation? (5 points)
   Justify your answer. (5 points)

3. What is wrong with the following solution to the mutual exclusion problem? (10 points to tell me when and how it fails)

   //Global variables
   int can_go;
   int request[2] = {0, 0};

   void enter_region(int pid) {
       can_go = pid; // set tiebreaker
       request(pid] = 1;
       while (request[1-] pid] & can_go != pid);
   } // enter_region

   void leave_region(int pid) {
       request[pid] = 0;
   } // leave_region
4. An interstate bus that can carry 50 passengers, has a single door that let one passenger get in or out at any time. Add semaphores to the following two functions to ensure that the bus will never be overloaded and (b) passengers will not collide with each other when embarking or debarking the bus. (20 points minus 2 points if you forget to initialize the semaphores)

```c
semaphore bus = _____;
semaphore door = _____;
embark()
{
    __________________;
    __________________;
    get_in();
    __________________;
    __________________;
} //embark

debark()
{
    __________________;
    __________________;
    get_out();
    __________________;
    __________________;
} //debark
```

5. Give one example of **private ports** in the UNIX system. (5 easy points)

6. How does the VMS scheduler handle processes that return to the ready state after completion of their I/O request? (15 easy points)