|  |  |  |
| --- | --- | --- |
| 1. | True/False | |
|  | Q: | T/F: A counting, or linear, barrier is the only barrier implementation required since other methods give no significant performance gains over it (not that performance is really an issue – it is a *barrier* after all). |
|  |  |  |
|  | A: | False, tree and butterfly implementations are both superior and result in less meaningless waiting time for barriers. |
|  |  |  |
|  | Ref: | Page 167 |

|  |  |  |
| --- | --- | --- |
| 2. | Multiple Choice | |
|  | Q: | Which of the following is not a type of synchronization barrier scheme:   1. Butterfly barrier 2. Counting barrier 3. Gatekeeper barrier 4. Tree barrier |
|  |  |  |
|  | A: | C. This option is made up. The other synchronization barrier types are all discussed in the text. |
|  |  |  |
|  | Ref: | Pages 165-169 |

|  |  |  |
| --- | --- | --- |
| 3. | Fill in the Blank | |
|  | Q: | Bidirectional data transfers are necessary in synchronization or else we risk \_\_\_\_\_\_\_\_\_\_. |
|  |  |  |
|  | A: | Deadlock. If two processes use blocking sends or other methods that preempt some sort of bidirectional transfer, deadlock can occur because both processes will be waiting for the reception of their message. |
|  |  |  |
|  | Ref: | Page 171 |

|  |  |  |
| --- | --- | --- |
| 4. | Short Answer/Code | |
|  | Q: | Differentiate between global synchronization and local synchronization. |
|  |  |  |
|  | A: | Global synchronization halts all processes/threads, while local synchronization only halts a select group of processes. Global synchronization is useful for things like coordinating timing or when all processes must stay in exact step with one another, while local synchronization might be useful when working with meshes or piplelines. |
|  |  |  |
|  | Ref: | Page 169 |