Exam 6

Question 1-

a. Mutual exclusion:
Either if process 0 or 1 are interested to enter in the CS, he must set the variable Q[i] as true.
Then, even if one of them enter in the CS, the second will not enter by examining the code:
$Read_i(Q[j] = T) \rightarrow$ blocked in the entry code.

b. Deadlock free:
The algorithm doesn't provide deadlock freedom as the next scenario shows:
Beginning with the process 0:
$Write_0(Q[0] = T) \rightarrow Write_0(R[0] = 0)$
Then, process 1 gets running time:
$Write_1(Q[1] = T)$
Then, process 0 gets running time:
$Read_0(Q[1] = T \&\& R[0] = R[1] = 0) \rightarrow$ blocking by the until statement, the process must go to line 1.
Then, process 1 gets running time:
$Write_1(R[1] = (0 + 1) \% 2 = 1) \rightarrow Read_1(Q[0] == T \&\& R[1] = (R[0] + 1) \% 2 = 1) \rightarrow$ blocking by the until statement, the process must go to line 1.
repeating this scenario...

c. Deadlock free:
The algorithm doesn't provide deadlock freedom as the next scenario shows:
Beginning with the process 0:
$Write_0(Q[0] = T) \rightarrow Write_0(Choose[0] = T) \rightarrow Write_0(R[0] == 0)$
Then, process 1 gets running time:
$Write_1(Q[1] = T)$
Then, process 0 gets running time:
$Write_0(Choose[0] = F) \rightarrow Read_0(Q[1] = T \&\& Choose[1] = F \&\& R[0] = R[1] = 0) \rightarrow$ blocking by the until statement, the process must go to line 1.
Then, process 1 gets running time:
$Write_1(Choose[1] = T) \rightarrow Write_1(R[1] = (0 + 1) \% 2 = 1) \rightarrow Write_1(Choose[1] = F) \rightarrow Read_1(Q[0] = T \&\& Choose[0] = F \&\& R[1] = (R[0] + 1) \% 2 = 1) \rightarrow$ blocking by the until statement, the process must go to line 1.
repeating this scenario...

d.
The algorithm doesn't provide deadlock freedom as the next scenario shows:
Beginning with the process 0:
$Write_0(Q[0] = T) \rightarrow Write_0(R[0] == 0) \rightarrow Write_0(Choose[0] = T)$
Then, process 1 gets running time:
$Write_1(Q[1] = T)$
Then, process 0 gets running time:
$Read_0(Q[1] = T \&\& Choose[1] = F \&\& R[0] = R[1] = 0) \rightarrow$ blocking by the until statement, the process must go to line 1.
From now only process 0 gets running time.
repeating this scenario…

Question 2-

b.
Shared :
semaphore deleter_thread_bc_of_seach = 1
semaphore deleter_thread_bc_of_insert = 1
int count_searcher = 0

code for Searcher thread:
<entry code>
count_searcher++
if (count_searcher == 1)
        down(deleter_thread_bc_of_seach)
<CS>
<exit code>
count_searcher--
if(count_searcher == 0)
        up(deleter_thread_bc_of_seach)

code for Inserted thread:
<entry code>
down(deleter_thread_bc_of_insert)
<CS>
<exit code>
up(inserted_thread_bc_of_insert)

code for Deleter thread:
<entry code>
down(deleter_thread_bc_of_insert)
down(deleter_thread_bc_of_search)
<CS>
<exit code>
up(inserted_thread_bc_of_search)
up(inserted_thread_bc_of_insert)