1. 1. \_declare\_box (mympool, 12, 20);
   2. \_init\_box(&mympool, 20, 12);
   3. rt\_alloc\_box
   4. Rt\_free\_box
2. Return value upon completion of a wait. These would be return codes like OS\_R\_OK.
3. 1. Inserts a p\_task into a priority queue ordered by descending priority
   2. Get task at head of list: it is the task with highest priority.
4. 1. Block running task and choose next ready task using rt\_get\_first. Delays the running task by a timeout (param) and sets its state to block\_state (param).
   2. Dispatch next task if any identified or dispatch highest ready task. If the next task is NULL, get and switch to the highest priority task in the pri queue. If the next task is non-NULL, check its priority compared to the head of the pri queue. If it is greater, run the next task. Else run the head of the pri queue and insert the next task into the pri queue.
5. rt\_mbx\_wait is only used for the receiving task to block it while it waits for a message. The OS\_R\_MBX flag is set by the sending task at the assembly level during context switching and that’s why it’s not in rt\_mbx\_wait function.
6. They use the OS\_XCB struct

void \*rt\_alloc\_box\_s (void \*p\_mpool) {

void \*ptr;

p\_TCB ptask;

int task\_id;

ptr = rt\_alloc\_box(p\_mpool);

if (ptr == NULL) {

task\_id = tr\_task\_self();

ptask = os\_active\_TCB[task\_id - 1];

rt\_put\_prio(&queue, p\_task);

return NULL;

}

else

return ptr;

}