# COMP3500: Midterm Exam 2 – Exercises

**Exercise 1 (Plickers):** When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place, is called critical condition.

**Exercise 2 (Plickers):** A condition variable is basically a container of threads that are waiting for a certain condition.

**Exercise 3 (Plickers):** Deadlock will arise if a process holding at least one resource is waiting to acquire additional resources held by other processes.

**Exercise 4 (Plickers):** If a resource allocation graph contains no cycles, then there is no deadlock.

**Exercise 5 (Plickers):** We can prevent the circular wait condition by

A) using a safe sequence B) using mutual exclusion

C) defining a linear ordering of resource types D) all of the mentioned

**Exercise 6 (Plickers):** Given a priori information about the \_\_\_\_\_\_\_\_ number of resources of each type that maybe requested for each process, we can construct an algorithm to guarantee that the system will not enter a deadlock state.

A) maximum B) minimum C) average D) approximate

**Exercise 7 (Plickers):** Processes, residing in main memory, are ready and wait to execute. The processes are kept on a list called:

A) job queue B) waiting queue C) ready queue D) process queue

**Exercise 8 (Plickers):** Which statement about the round robin scheduling algorithm in a time shared system is correct?

A) using very small time slices converts it into Shortest Job First algorithm

B) using very small time slices converts it into First come First served scheduling algorithm

C) using extremely small time slices increases performance

D) using very large time slices converts it into First come First served scheduling algorithm

**Exercise 9 (Plickers):** CPU fetches the instruction from memory according to the value of instruction register.

**Exercise 10 (Plickers):** What type of address do program always deals with?

A) absolute address B) logical address C) physical address D) relative address

**Exercise 11:** There are three processes accessing four resources labeled from R1 to R5. Please use a resource allocation graph to detect if there is a possible deadlock.

|  |  |  |
| --- | --- | --- |
| void P1() {  while (true) {  lock(R2);  lock(R1);  lock(R3);  /\*Critical region\*/  unlock(R2);  unlock(R1);  unlock(R3);  }  } | void P2() {  while (true) {  lock(R5);  lock(R3);  lock(R4);  /\*Critical region\*/  unlock (R5);  unlock (R3);  unlock (R4);  }  } | void P3() {  while (true) {  lock(R4);  lock(R1);  /\*Critical region\*/  unlock (R4);  unlock (R1);  }  } |

**Exercise 12:** The following four processes are scheduled by the deadline-driven scheduler, which give high priorities to processes with the earliest deadlines. We assume that a newly arrived process with a loose deadline.

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival Time | CPU-burst Time | Deadline |
| P1 | 0 | 18 | 30 |
| P2 | 8 | 12 | 22 |
| P3 | 12 | 7 | 20 |
| P4 | 23 | 5 | 28 |

1. Draw a Gantt chart illustrating the execution of these.
2. Which processes miss their deadlines?
3. What are the turnaround times of the four processes?
4. What are the waiting times of the four processes?