COMP 3500: Homework 2

Mary Mitchell
mem0250

Points Possible: 100.
Note: You do not need to submit hard copies.

There should be no collaboration among students. A student shouldn't share any project code with any other student. Collaborations among students in any form will be treated as a serious violation of the University's academic integrity code.

Goals:

- To understand the principles of deadlocks.
- To learn how to solve deadlock and starvation problems.
- To collaborate and discuss deadlock problems with your group members.

Questions:

1. [40 points]

In the code below, three processes are competing for six resources labeled A to F.

a. Using a resource allocation graph (Figures 6.5 and 6.6), show the possibility of a deadlock in this implementation.

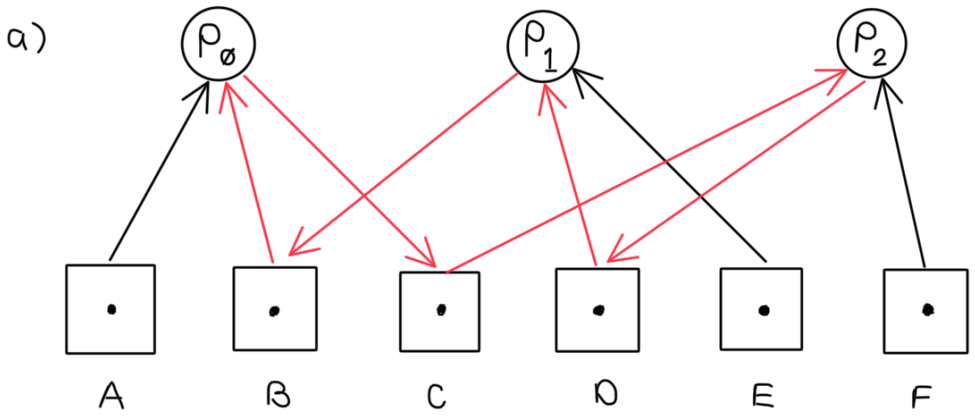
b. Modify the order of some of the get requests to prevent the possibility of any deadlock. You cannot move requests across procedures, only change the order inside each procedure. Use a resource allocation graph to justify your answer.

```
void P0()
{
  while (true) {
    get(A);
    get(B);
    get(C);
    // critical region:
    // use A, B, C
    release(A);
    release(B);
    release(C);
  }
}
```

```
void P1()
{
  while (true) {
    get(D);
    get(E);
    get(B);
    // critical region:
    // use D, E, B
    release(D);
    release(E);
    release(B);
  }
}
```

```
void P2()
{
  while (true) {
    get(C);
    get(F);
    get(D);
    // critical region:
    // use C, F, D
    release(C);
    release(F);
    release(D);
  }
}
```

a)

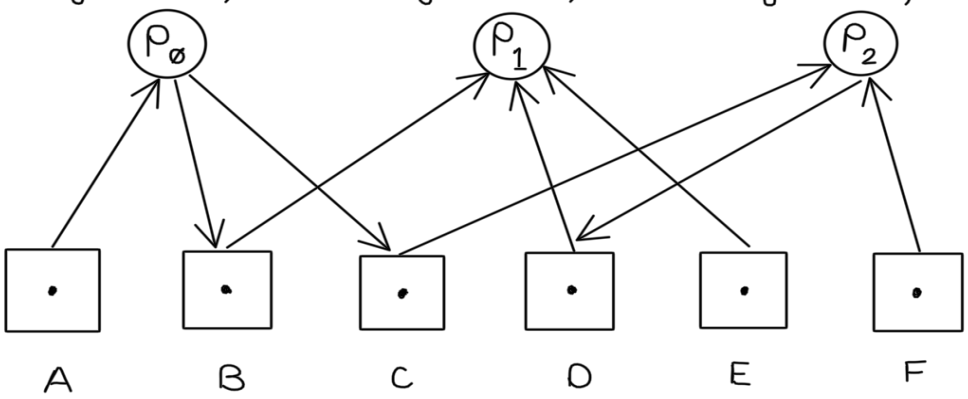


P_0 : get(A)
P_1 : get(D)
P_2 : get(C)
P_0 : get(B)
P_1 : get(E)
P_2 : get(F)
P_0 : get(C)
P_1 : get(B)
P_2 : get(D)

This resource allocation graph was created by executing a single instruction from a process and then moving to the next process and doing the same, starting at P_0, then P_1, then P_2, and back to P_0 to continue the execution. The red lines show the cycle in which deadlock has occurred. This is an instance of circular wait.

b) To prevent circular wait, we can impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration. To do this each process will request its resources in alphabetical order. Like part a, this resource allocation graph was created by executing one line from a process before rotating to the next process.

```
Void P0()        Void P1()        Void P2()
  get(A);          get(B);          get(C);
  get(B);          get(D);          get(D);
  get(C);          get(E);          get(F);
```



2. [20 points]
Suppose the following two processes, `foo` and `bar` are executed concurrently and share the semaphore variables `S` and `R` (each initialized to 1) and the integer variable `x` (initialized to 0).

```
void foo( ) {
   do {
     semWait(S);
     semWait(R);
     x++;
     semSignal(S);
     SemSignal(R);
   } while (1);
}
```

```
void bar( ) {
do {
    semWait(R);
    semWait(S);
    x--;
    semSignal(S;
    SemSignal(R);
} while (1);
}
```

Can the concurrent execution of these two processes result in one or both being blocked forever? If your answer is yes, please give an execution sequence in which one or both are blocked forever.

These two processes can result in both being blocked forever. The following execution would deadlock them both in a circular wait.

① foo : semWait(S); (will immediately get resource)
bar : semWait(R); (will immediately get resource)
foo : semWait(R); (will have to wait on bar)
bar : semWait(S); (will have to wait on foo)

Once these executions complete they will both be blocked from executing any further, as they will each be waiting on the other to release their resources.

3. [20 points]
What is the difference among deadlock avoidance, detection, and prevention?

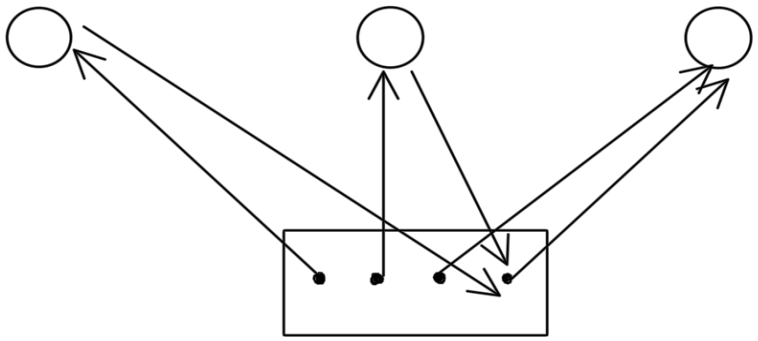
Deadlock prevention provides a set of methods to ensure that at least one of the necessary deadlock conditions cannot hold. This does not require the system to have any additional information on how resources are used. Instead this solution limits how requests can be made. As such it is a static solution.

Deadlock avoidance is a solution in which the system makes the appropriate dynamic choices based on the current state of resource allocation. This requires the system to have additional information about how resources are to be requested unlike deadlock prevention. Based on this information the system decides whether each request will result in a safe or unsafe state. This makes the solution a dynamic one.

Deadlock detection, unlike the previous two solutions, waits for deadlock to occur to implement its solution. In this situation the system must determine whether deadlock has occurred and provide an algorithm to recover from it.

4. [20 points]
Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock-free.

To be in deadlock, each process would need to be holding a resource and waiting on another. However, if each of the three processes are holding a resource, since there are four resources there will always be another resource available for one of the processes to obtain and complete their tasks. Essentially, even if all the processes request resources at once, with four resources, one process will always obtain two resources, preventing deadlock.



Submission:

- A heading at the top of your file contains your name and your Auburn UserIDs.
- Submit your solution as a single PDF file named as "hw2.pdf" through Canvas
- File formats other than PDF will not be accepted by Canvas.

Late Submission Penalty:

- Ten percent (10%) penalty per day for late submission. For example, an assignment submitted after the deadline but up to 1 day (24 hours) late can achieve a maximum of 90% of points allocated for the assignment. An assignment submitted after the deadline but up to 2 days (48 hours) late can achieve a maximum of 80% of points allocated for the assignment.
- Assignment submitted more than 3 days (72 hours) after the deadline will not be graded.

Rebuttal period:

- You will be given a period of one week (i.e., 7 days) to read and respond to the comments and grades of your homework or project assignment. The TA may use this opportunity to address any concern and question you have. The TA also may ask for additional information from you regarding your homework or project.