Spring 2019 **H. Hakimzadeh**	**C435 Operating Systems** **ULTIMA 2.0**	**30 points** **Due Monday Feb 25** **11:59pm**

ULTIMA is a simulated cooperative multitasking operating system. Currently this system lacks many necessary operating system components. Your job for the remainder of this semester is to complete each component and develop a fully functional multi-tasking environment.

PHASE 1: Scheduler and Semaphore

The current system lacks any kind of process or resource synchronization. This problem is clear when two processes try to use a shared resource such as a printer. In general, the operating system must provide a mechanism for controlling access to any shared resource, such as printer, memory or disk.

Your job during this phase is to implement SEMAPHORES. You must be able to support any number of semaphores. This is critical, since future phases of this project introduce new resources which will certainly need synchronization.

In addition to implementing the semaphore, you also need process table (more accurately a thread table). The thread table should be implemented dynamically for example a linked list of Task Control Block's (TCB). At this point, the TCB's are needed to preserve the STATE of each Task. (i.e. RUNNING, READY, BLOCKED, DEAD). In the future phases, TCB's will maintain other information such as a pointer to memory allocated by each task, file descriptor, and task priority.

One way to implement the Process Table is to create a TCB for each task when it is being created and insert the task ID, name, and other relevant information in the TCB. Then insert the TCB in a scheduling ring or linked list.

```
int Create_Task(char *task_name, etc...);      // return the T_id
```

In short, we need at least two classes. One that implements a scheduling ring and the associated algorithms and the other for implementing semaphores.

You must develop clear and concise test cases to show how and why your algorithms work. Plan these cases as a group. Each class that you implement must contain one or more debugging functions which print the current state of that class.

```
Class scheduler {
        TCB *process_table;
        create_task();          // create appropriate data structures and calls coroutine()
        destroy task();         // to kill a task (Set its status to DEAD)
        yield();                // strict round robin process switch.
        dump(int level);        // debugging function with level indicating the verbosity of the dump
                                include some functions which will allow you to dump the contents of the
                                process table in a readable format.  See the expected output section
                                (below) for suggestions.
        garbage_collect();      // remove dead task, free their resources, etc.
}
```

```
Class semaphore {
        char resource_name [64];    // the name of the resource being managed
        int sema_value;             // 0 or 1 in the case of a binary semaphore
        queue *sema_queue;
        down();                     // get the resource or get queued!
        up()                        // release the resource

        dump(int level);            // include some functions which will allow you to dump the
                                    contents of the semaphore in a readable format.  See the expected
                                    output section (below) for suggestions.
}
```

Suggestions and Hints:

- Do not use a busy wait in Down (), instead block the task.
- Create a STATE parameter in the Task Control block (TCB) and have the scheduler check to see if a process is ready, blocked, or dead before scheduling.
- Be aware of deadlocks.

Expected Output:

The output from this phase must clearly show that the scheduler and process table work properly. In addition, you should design a test scenario which clearly shows that the semaphore is performing its job correctly. I suggest having two or three tasks try to acquire a resource (say write to a window) and show that one of them will get the resource and the others promptly get queued and their state changes to BLOCKed.

Sample Process Table Dump:

In order to show that your scheduler and process table operate correctly, develop a public method called dump() which simply produces a table similar to the one below:

Task Name	Task ID	State	etc.
Task1	1	Ready	
Task2	2	Running	
Task3	3	Ready	

Sample Semaphore Dump:

To show that your semaphore is working properly you should use your dump() function which shows the sema-value as well as the content of sema-queue:

```
Semaphore Dump:
Resource:     "Resource Name"          // i.e. printer, memory, etc.
Sema_value:   0                        // 0 or 1
Sema_queue:   T-id-->T-id -->....
```

What to hand in:

- Each group should put together two binders (one for you and one for me) with the following information:
  - Cover page with your name, course # and name, project name, date, group member names, e-mails, etc.
  - Project Abstract (You will add this abstract with every new phase)
  - Table of content for the Project.
  - Resume from each group member.
  - For each phase of the project include the following:
    - See below for **"Guideline for each phase of the project"**
    - Self and Peer evaluation form. (Fill them out but do not include it in your binder.) These are anonymous and should be returned to me directly.

Guideline for each phase of the project

ULTIMA 2.0

Each group should put together a binder with the following information:

1) Cover Page

ULTIMA 2.0
CS-435 Operating System
Phase # - Phase Name

By:

Team Members Up to Two Members
Team Name (if any)
(Names and e-mails)

Date

2) Phase Abstract	(Short half a page abstract)
5) Phase Description	(External Documentation)
6) Design Diagrams	(Design diagrams. including the data structures and class hierarchy, nicely drawn using a drawing package)

7) Source Code

- Internal Documentation (Each function, Class, Struct, etc. should be documented)
- Nicely formatted (Use a smaller font when printing your source code so it does not wrap around.)
- Copy of the make file. and/or dependency graph of modules and files
- ".h", " ".cpp" , makefile (Separated by Modules)
    - Ultima.h, Ultima.cpp
    - Sema.h, Sema.cpp
    - Sched.h, Sched.cpp
    - Mem_mgr.h, Mem_mgr.cpp

8) Output (Must be coherent and annotated)

- Produce a test plan and include it in your output.
- Explain how and why your system is working.
- Explain what objectives your system should meet in order to be evaluated as a correctly functioning phase.
- Readable Dump from various data structures (semaphore, scheduler, memory manager, message handler, file system, and print spooler.)
- Screen dumps (nicely formatted and annotated by hand)

Self / Peer Evaluation Form

C435 - Operating Systems
ULTIMA 2.0

Date: __________ Phase Number: ___ Phase Title: _______________

Your Name: _____________________

1) Given the following categories, evaluate yourself and your group member. Estimate the contributions base both on hours and percentage of total work. *(For example: attended all the meeting 100%, 4 hours, I implemented a small class, but the rest of the classes were implemented by my team member, so I estimate my contribution as 25% and 7 hours, I did most the documentation, so my contribution was 90% and spent 3 hour)*

a) Attending group meetings:	%____	Hrs:___
b) Participation in the design:	%____	Hrs:___
c) Participation in the implementation:	%____	Hrs:___
d) Participation in the debugging:	%____	Hrs:___
e) Participation in testing:	%____	Hrs:___
f) Participation in the documentation:	%____	Hrs:___
g) Overall contribution to the group:	%____	Hrs:___

h) Other Activities (Explain): ____

Your Team Member's Name: _____________________

a) Attending group meetings:	%____	Hrs:___
b) Participation in the design:	%____	Hrs:___
c) Participation in the implementation:	%____	Hrs:___
d) Participation in the debugging:	%____	Hrs:___
e) Participation in testing:	%____	Hrs:___
f) Participation in the documentation:	%____	Hrs:___
g) Overall contribution to the group:	%____	Hrs:___

h) Other Activities (Explain): ____

2) What were your exact responsibilities for this phase?

A) Design Activity: ☐ Data structure ☐ Algorithms
Name the data structures and algorithms you worked on:

B) Implementation Activity: What functions/classes did you work on?

C) Documentation Activity: ☐ internal ☐ external

Name of functions/classes you documented:

D) Testing Activity: ☐ modules ☐ overall system

Name of functions you tested:

E) Other Activities: ____

3) How would you improve the system you have implemented? (Be specific)

4) Do you understand the design decisions that were made by your group?
If so explain:

If not explain:

Do you agree with this design decisions?	☐ Yes	☐ No
If yes did you argue for this design?	☐ Yes	☐ No
If no did you argue against this design?	☐ Yes	☐ No

5) What did you learn during this phase of the project?

6) Which of the following helped you to understand and complete this phase of the project? (Check all that apply)

☐ Text Book	☐ Class Discussion
☐ Class notes	☐ Group Discussion
☐ External Sources (Specify)______________	