CMPUT 379 Assignment 3, Fall 2014

Due Monday, Nov 24 (submit by 11:55 pm)

Worth 20 marks (however, 5% of total course weight)
(Deadlock management, Banker's algorithm)

Objective

In this assignment you will simulate Banker's algorithm in a multi-processor computing environment. Banker's algorithm is used to avoid deadlocks in a computing environment.

Input

You would define the number and names of different types of resources available in system as well as the number of instances each resource type has. CPU should be one type of resource which may have more than one instance. You would also define the number and names of the processes in the system as well as their resource requirements. You would define processes at the beginning of your simulation. So, you need to mention the arrival time of each process (this is simulation time, not the system time, and you start simulation at simulation time 0) at the beginning. You would also mention the total length of execution time of each process. An example of input may be

```
?> Number of different resource types: 5
?> Names of each resource type: CPU A B C D
?> Number of instances of each resource type: 2 3 5 2 8
```

Above data mean that the computing system has 2 instances of CPU, 3 instances of A, 5 instances of B, 2 instances of C and 8 instances of D.

```
?> Number of processes: 5
?> Details of process-1: P1 2 2 1 1 3 10 7
```

Above data mention that process-1 has the name P1. During its execution it requires 2 instances of CPU, 2 instances of A, 1 instance of B, 1 instance of C and 3 instances of D. The process would start at simulation time 10 and would have a total execution length of 7

simulation time units. For, simplicity you may only consider integer values for process starting time and process execution time.

In this scenario, you must also have to provide details of process-2, process-3, process-4 and process-5 in the same way.

Output

All of your output would go to the screen. You must output the status of the system (as shown below) each time an event occurs. An event may be the starting or ending of the simulation, the arrival or departure of a process. System status would include the description of the current event as well as the status of all processes available in the system at that time. For example, when process P1 finishes its execution you would output the following message.

```
?> Simulation time: 17
?> Process P1 has just been finished.
```

When process P3 starts execution you would output the following message.

```
?> Simulation time: 20
?> Process P2 is running.
?> Process P3 has just started execution.
```

When a process arrives you would output the following message.

```
?> Simulation time: 22
?> Process P2 is running.
?> Process P3 is running.
?> Process P4 has just arrived at the system.
```

When the last process finishes its execution you would output the following message.

```
?> Simulation time: 38
```

```
?> Process P5 has just been finished.

?> No process is available for execution, system is idle.

?> Simulation is ended.
```

If no process is running in the system and available resources are not sufficient to start any process that exists in the system, you would output the following message.

```
?> Simulation time: 28

?> Process P3 has just been finished.

?> Process P4 is idle.

?> Process P5 is idle.

?> No process is running. Available resources are not sufficient to run any idle process or to avoid deadlocks. System is halted.

?> Simulation is ended.
```

Marking

This assignment will be marked out of 20 marks.

You are expected to deliver a good quality code, which is easy to read and comprehend. **The quality and style of your coding will be marked.**

- 15 marks – implementation of all features
- 5 marks – quality of coding, bug free program

Deliverables

You must deliver a single tar file that contains a single code-file as well as a `Makefile` to the build application. **Any different submission would be penalized**.

Mandatory Coding Style

Consistent and readable C coding style is *very* important to spotting bugs and having your code readable by others. You **MUST** comply with following coding style:

- Proper indentation should be included

- TAB must be used for indentation
- NO line may be longer than 80 chars
- All functions must be prototyped
- All variables must be declared at the start of a block
- C style comments must be used (not C++ style)

Important: please always remember to remove from your machine the running processes before logging off. Generally, make sure there are no stray processes left on your machine before leaving.