# Homework 4 CSC246

Respond to each homework 4 question and submit your work electronically using the course web site. Please submit your non-program answers in either plain text or Microsoft Word documents (or rtf). For the programming question, submit the source file.

For this homework, the 4th problem can be answered using either C/C++ pthreads or Java threads.

If you need help, please contact the TA at once and arrange to get help. The TA's email address is on the syllabus.  
  
This homework is due on the date mentioned on the main website **by midnight.** This homework contains 100 points and is 10% of your total course grade.

*5 points*

## Question 1: Sockets

Investigate and explain how sockets are used for ftp. What ports are used for the service? What functionality is associated with each of these ports?

Consult <http://www.deskshare.com/resources/articles/ftp-how-to.aspx>

Sockets:

A socket is an endpoint of a two-way communication. It is concatenation of IP address and port.

To establish the connection of the communication, the client and server will need get the IP address and the port from the socket.

Ports:

20: data port, used for transferring data

21: command port, send commands

*30 points*

## Question 2: CPU Scheduling

Four batch jobs P1 through P4 are to be scheduled using various algorithms. Their next CPU burst estimates are 8, 3, 7 and 5 units.

For each of the following algorithms, fill in the following table. Ignore context switching overhead.

You do not have to submit the Gantt chart, just the completed table. However, it is OK to submit the Gantt chart.

### Algorithms:

* First Come First Served
* Shortest Job First (non-preemptive)
* Round Robin (Quantum = 5)

### First Come First Served

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PID | TimeIn | TimeOut | Turnaround Time | CPU Needs | Total Wait Time |
| P1 | 0 | 8 | 8 | 8 | 0 |
| P2 | 1 | 11 | 10 | 3 | 7 |
| P3 | 2 | 18 | 16 | 7 | 9 |
| P4 | 3 | 23 | 20 | 5 | 15 |

**AVG TA = 13.5**

**AVG Wait = 7.75**

### Shortest Job First (non-preemptive)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PID | TimeIn | TimeOut | Turnaround Time | CPU Needs | Total Wait Time |
| P1 | 0 | 8 | 8 | 8 | 0 |
| P2 | 1 | 11 | 10 | 3 | 7 |
| P3 | 2 | 23 | 21 | 7 | 14 |
| P4 | 3 | 16 | 13 | 5 | 8 |

**AVG TA = 13**

**AVG Wait = 7.25**

### Round Robin Quantum = 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PID | TimeIn | TimeOut | Turnaround Time | CPU Needs | Total Wait Time |
| P1 | 0 | 21 | 21 | 8 | 13 |
| P2 | 1 | 8 | 7 | 3 | 4 |
| P3 | 2 | 23 | 21 | 7 | 14 |
| P4 | 3 | 18 | 15 | 5 | 10 |

**AVG TA = 16**

**AVG Wait = 10.25**

*10 points*

## Question 3 Predicting Next Burst for SJF

A given system is using alpha = 1/2 to predict run times. The first guess was 100 and the actual run times were 40, 70, and 120 units of time. What is the prediction for the next burst time? Show your work.

First guess = 100, actual = 40

Next Guess: T1 = ½ (40) + ½ (100) = 70

Next Guess: T2 = ½ (70) + ½ (70) = 70

Next Guess: T3 = ½ (70) + ½ (120) = 95

*45 points*

## Question 4: The Game of Life in a Threaded World

Write a multithreaded Java or pthreads program that simulates the "Game of Life." The simulation occurs on an M-by-N rectangular array of cells. Each cell has only two values: 0 and 1. Each cell has eight neighbors: up, down, left, right, and four diagonally. Cells on the four edges of the grid still have 8 neighbors…they just wrap around. If the grid is declared as:

**int [] [] grid = new int [M] [N];**

then the neighbors of an interior cell **grid[i][j]** are **grid[i-1][j], grid[i+1][j], grid[i][j-1], grid[i][j+1], grid[i-1][j-1], grid[i-1][j+1], grid[i+1][j-1],** and **grid[i+1][j+1]**.

The game works like this. Fill the grid with initial values. Then for each cell in the grid, compute the value for this cell in the next generation (stored in a separate grid). All changes occur simultaneously, called the generation.

 A 1 cell value stays 1 if exactly two or three neighbors are 1 valued.

 A 1 cell value becomes 0 if less than two or greater than three neighbors are 1 valued.

 A 0 cell value becomes 1 if exactly three neighbors are 1 valued.

 A 0 cell value stays 0 if less than three or greater than three neighbors are 1 valued.

Design a separate thread to calculate the new value for a specific cell at position (i,j).

In pseudo-code:

**Create an M x N grid for the starting state and populate the array.**

**Create an M x N grid for the next generation state.**

**Display starting state grid.**

**Loop for specific number of generations:**

**Create M x N threads, providing each thread with its (i,j) position**

**in the starting state grid. [For java programmers, you must**

**make the threads schedulable by invoking start.]**

**Join all threads. <-must be a separate loop from creation loop**

**Display the next generation grid.**

**Make the starting state grid = next generation grid.**

Do *not* create and join each thread inside each loop. This results in no parallelism.

The input data for your program will consist of a file with the values of M and N followed by the values for the starting state of each cell in the rectangular grid.

The first line of the file will be two integer values, the number of rows followed by the number of columns.

For example,

5 4

The following lines contain sequences of 0's and 1's for each row.

For example,

0 0 1 0

1 1 1 1

1 0 0 1

0 0 0 1

1 1 1 1

You may assume the files are formatted correctly.

There are 4 practice files available on the course web site:

 EveryoneDies.txt

 DiesMoreSlowly.txt

 StableState.txt

 Oscillates.txt

Submit your work as Life.c, Life.cpp, or Life.java (with a main for execution to begin).

Your program must accept two command-line arguments. The first is the input file name and the second is the number of generations.

The program must display the initial rectangular grid, one row per line, and the grid after each generation. Again, display the grid one row per line.

There is a reference for using shared data among java threads on the course web site called **SharedDataExample.java**.

For pthreads, make the before and after grids global variables. There are several code examples of pthreads in the Resource link of the course website. The **lifethreadexample.c** code on the course web site demonstrates how to pass two into to a thread as one argument.

*10 points*

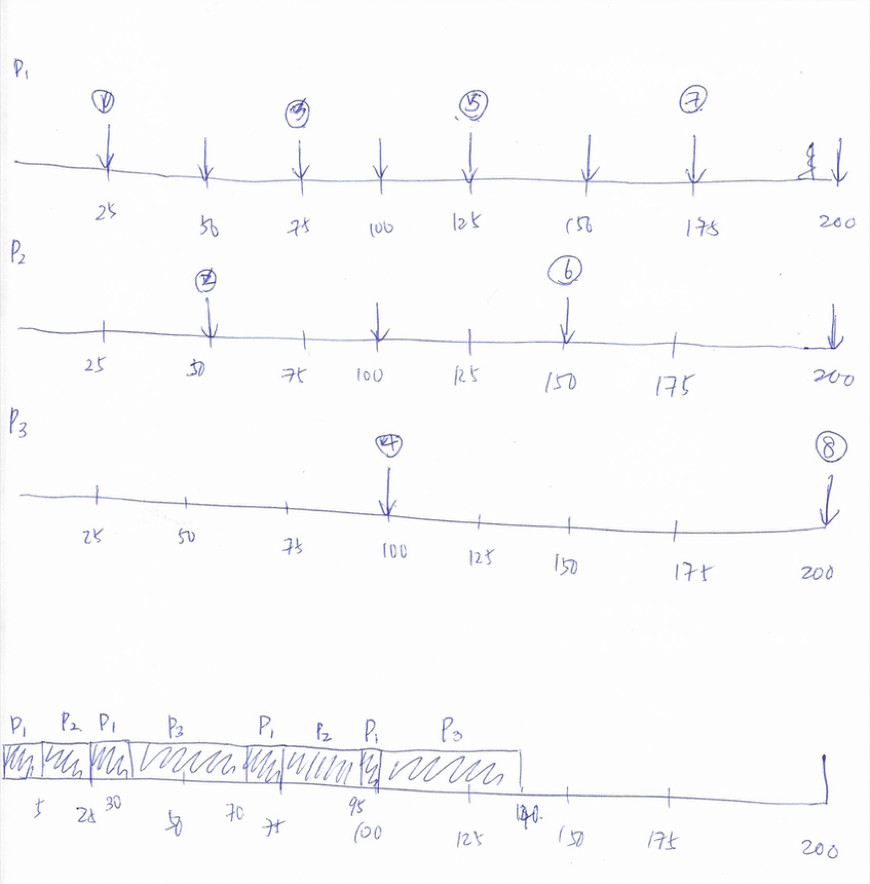
## Question 5: CPU Scheduling using EDF

Using a Gantt chart, show how the following periodic tasks would be scheduled for the first 200 units of time using the Earliest Deadline First scheduling algorithm.

**Task 1 Burst Size = 5 Period = 25**

**Task 2 Burst Size = 20 Period = 50**

**Task 3 Burst Size = 40 Period = 100**

****