CS 350 Homework 6 Tim Duffy

1) Consider the following specification of three periodic tasks:

	Period	**CPU time / Period**
Task 1	T seconds	1 seconds
Task 2	5 seconds	1 second
Task 3	7 seconds	2 seconds

1. What is the minimum value of T that would result in a schedulable system using Rate Monotonic Scheduling?

   RMS is schedulable if the sum of each Ci/Ti <= N(2^(1/N) - 1)
   (2/7 + 1/5 + 1/T) <= 3(2^(1/3) - 1)
   0.485714286 + 1/T = 0.77976315 - removing the less than will give us the minimum
   1/T = 0.294048864 - value of T that will work

   T = 3.4 seconds

2. Show the first 21 seconds of execution of the system above using the value of T that you obtained in part (a) above. Assume that all tasks are started at time 0 and that RMS scheduling is used.

t = 0:	Task 1 begins (smallest period = highest priority)
t = 1:	Task 2 begins
t = 2:	Task 3 begins
t = 3.4:	Task 3 is suspended (0.6 remain), Task 1 begins
t = 4.4:	Task 3 resumes
t = 5:	Task 2 begins
t = 6:	Idle!
t = 6.8:	Task 1 begins
t = 7.8:	Task 3 begins
t = 10:	Task 2 begins
t = 10.2:	Task 2 suspended (0.8 left), Task 1 begins
t = 11.2:	Task 2 resumes
t = 13.6:	Task 1 begins
t = 14.6:	Task 3 begins
t = 15:	Task 3 suspended (1.6 left), Task 2 begins
t = 16:	Task 3 resumes
t = 17:	Task 3 suspended (0.6 left), Task 1 begins
t = 18:	Task 3 resumes
t = 20:	Task 2 begins
t = 20.4:	Task 2 suspended (0.6 left), Task 1 begins
t = 21.4:	Task 2 resumes

3. What is the minimum value of T that would result in a schedulable system using Earliest Deadline First?

   RMS is schedulable if the sum of each Ci/Ti <= 1
   (2/7 + 1/5 + 1/T) <= 1
   0.485714286 + 1/T = 1
   1/T = 0.514285714

   T = **1.94 seconds**

4. Show the first 21 seconds of execution of the system above using the value of T that you obtained in part (c) above. Assume that all tasks are started at time 0 and that EDF scheduling is used.

1.94 seconds	1
5 seconds	1
7 seconds	2

t = 0.00:	Task 1 begins (deadline at 1.94)
t = 1.00:	Task 2 begins (deadline at 5.00)
t = 1.94:	Task 2 suspended (0.06 remaining), Task 1 begins (deadline at 3.88)
t = 2.94:	Task 2 resumes and completes
t = 3.00:	Task 3 begins (deadline at 7.00)
t = 3.88:	Task 3 suspended (1.12 remaining), Task 1 begins (deadline at 5.82)
t = 4.88:	Task 3 resumes and completes
t = 6.00:	Task 1 begins (deadline at 7.76)
t = 7.00:	Task 2 begins (deadline at 10.00)
t = 7.76:	Task 2 suspended (0.24 remaining), Task 1 begins (deadline at 9.7)
t = 8.76:	Task 2 resumes and completes
t = 9.00:	Task 3 begins (deadline at 14.00)
t = 9.70:	Task 3 suspended (1.30 remaining), Task 1 begins (deadline at 11.64)
t = 10.70:	Task 3 resumes
t = 11.64:	Task 3 suspended (0.36 remaining), Task 1 begins (deadline at 13.58)
t = 12.64:	Task 3 resumes and completes
t = 13.00:	Task 2 begins (deadline at 15)
t = 14.00:	Task 1 begins (deadline at 15.52)
t = 15.00:	Task 3 begins (deadline at 21)
t = 15.52:	Task 3 suspended (1.48 remaining), Task 1 begins (deadline at 17.46)
t = 16.52:	Task 2 begins (deadline at 20)
t = 17.46:	Task 2 suspended (0.06 remaining), Task 1 begins (deadline at 19.4)
t = 18.46:	Task 2 resumes and completes
t = 18.52:	Task 3 resumes and completes
t = 20.00:	Task 1 begins (deadline at 21.34)
t = 21.00:	Task 2 begins (deadline at 25)
t = 21.34:	Task 2 suspended (0.66 remaining), Task 1 begins (deadline at 23.28)

2) a. Show that starvation is possible in the above system. Specifically, describe a particular scenario under which starvation is possible.

Starvation is possible for requests for files that have been removed from the cache, and then are called again. Say the file with ID #1 is requested. At first, it will be placed in Q2, then the file will be stored in the cache. Eventually, it will be replaced by another file after the cache fills up.

Now, the next time the file #1 is requested, it will not be found in the cache, so the request for file #1 will be moved to Q2. This is when starvation occurs. Since everything currently in the cache is greater than #1, only requests for files greater than 1 will be taken from Q2 (according to rule ii). following the given rules, there is no way for this request to be served.

b. What kind of disk scheduling algorithm was your friend thinking of when he suggested the above fix?

He was thinking of the N-Step SCAN.

c. Discuss the effect of N on the effectiveness of the cache (i.e., is a large N better for improving the cache performance or is a small N better?)

A larger N will cause cache effectiveness to decrease. As N decreases, there is less of a chance that a request will go unfulfilled. The faster the file is moved to cache, the more likely it is to be in the cache when requested again. For example, say a file is needed every 5 seconds. If, as in part a, the process starves, any requests for this file will result in a cache miss. However, a smaller N will reduce the chance of starvation, improving cache performance.

d. Discuss the effect of N on the fairness of the above system (i.e., is a large N better for improving fairness or is a small N better for fairness?)

A smaller N will increase fairness to requests. As N decreases, fewer and fewer requests will be able to pass by any other requests. As a result, the worst case wait time for any request will decrease. If, for example, N=1, a process will never have to wait for another before it begins service. However, if N is large, then there is a greater chance of starvation, as explained in part a.

3. What did you expect the results to show? Did they match your expectations?

I expect the FCFS and Random scheduling processes will have similar response times and total head movement. I suspect that the SCAN scheduling process will trade off total head movement for response time.

My results back my expectations. They show that Random has the quickest response time by far, but has the most total head movement. SCAN certainly has the least total head movement, but averages the slowest response time. The response time for FCFS falls in the middle of the two others, and total head movement is similar to that of Random scheduling.