Lab Assignment 7:

Online Preemptive Priority Scheduling on a Single Processor:

Consider a processor which executes tasks. Each task is described by a 4-tuple (I, R, C, P), in which "I" is the task identifier. It is a unique integer ($I \geq 1$) which identifies a task. R is the release time of the task. "R" is an integer ($R \geq 0$) which specifies the time at which the task I is ready to run if it is allocated on the processor. The task I can be run at a time $T \geq R$.

"C" is the computation time of the task I. C is an integer ($C \geq 1$) which specifies the duration for which the task I will run. "P" specifies the priority of the task I. P is an integer ($P \geq 0$). At any given instance of time, only the task with highest priority can be run on the processor. Online scheduling means that the processor has knowledge of (at a time T) only the tasks with $R \leq T$. At any given instance of time T, the processor cannot know the "future" tasks with $R > T$. The scheduling decisions must be taken by considering the tasks with $R \leq T$. Preemptive scheduling means that if at any time a task arrives which is having higher priority than the currently running task, then the new task is allocated the processor on

Input : You will be given the 4-tuples (I, R, C, P) describing the tasks (one task per line) sorted in non-decreasing order of R:

$I_1\ R_1\ C_1\ P_1$
$I_2\ R_2\ C_2\ P_2$
$\vdots$
$I_n\ R_n\ C_n\ P_n$

All values are separated by blank space.

Output : For each unit of time you have to print the task that is running on the processor (separated by blank space). If the processor is idle (no task is running on the processor), then you have to print "0" (zero) for the corresponding time unit.

$T_1\ T_2\ T_3 \ldots T_m$

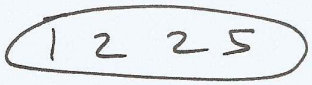
Task T_i runs for $i-1 < T \le i$, where $1 \le i \le m$. $T_i = 0$ means that the processor is idle for $i-1 < T \le i$. We can have $T_i = T_j$ for $i \ne j$ (a task can run for more than one unit of time).

Sample Input :

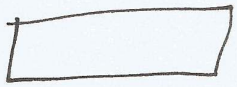
1 2 2 5
2 2 1 4
3 3 1 6
4 3 2 3

Procedure: You will have to implement the priority queue using a Binary Search Tree (BST). For inserting a node into the priority queue, implement the algorithm Tree-Insert for BST. For deleting a node from the priority queue, modify the algorithm Tree-Delete for BST so that it always deletes a node with highest key. Scan the input one line at a time. The processor is idle for $0 \le T \le 2$. At $T = 2$, we have two tasks (1 and 2). Initialize the BST to be empty at the start. Insert the tasks 1 and 2 in order into the BST.

T=2:

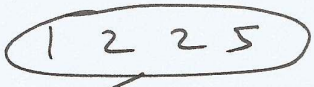


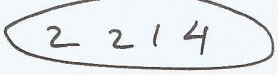
BST



Processor

T=2:



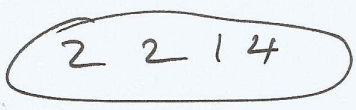


BST

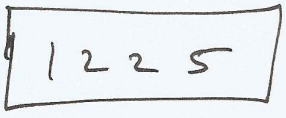
Processor

When all tasks are inserted into the BST at $T = 2$, delete the task with highest priority and allocate it on the processor.

T=2:



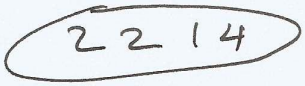
BST



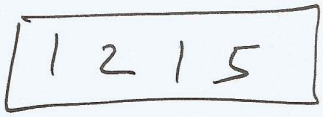
Processor

At T=3, the task 1 has run for 1 unit of time. Update its "C" value:

T=3:



BST

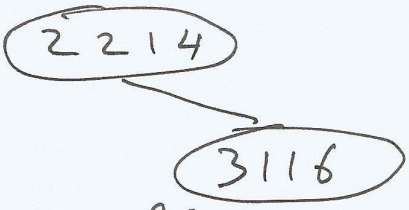


Processor

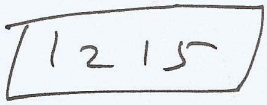
Now the task 3 arrives with priority 6. It is having higher priority than the currently running task. First insert 

in the BST:

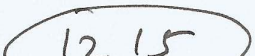
T=3:



BST

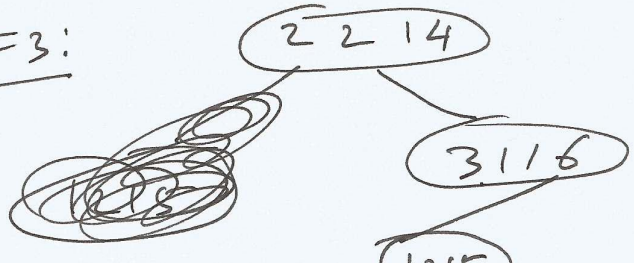


Processor

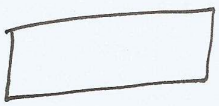
Now insert 

back in the BST because a new task with higher priority has arrived. If the new task is not having higher priority than the currently running task, then we will not put back the currently running task.

T=3:



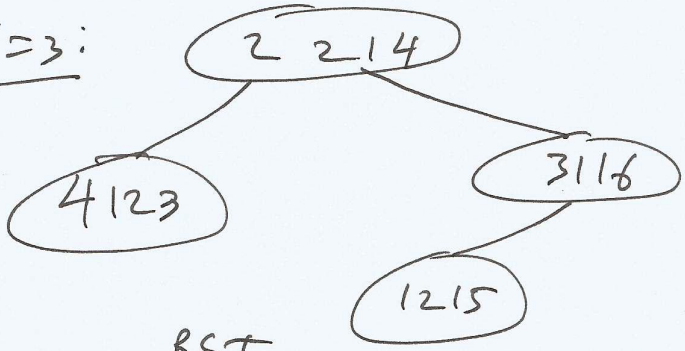
BST



Processor

Now another task (4, 1, 2, 3) arrives at T=3. Insert it into the BST:

T=3:

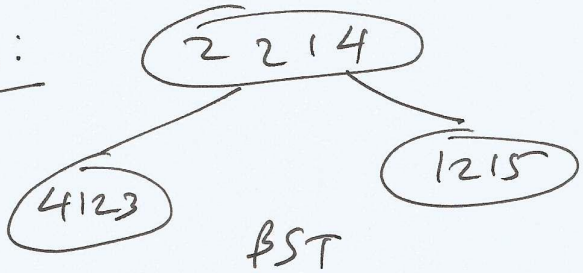


BST

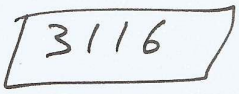
Processor

Now delete the highest priority task from the BST, and allocate it on the processor:

T=3:



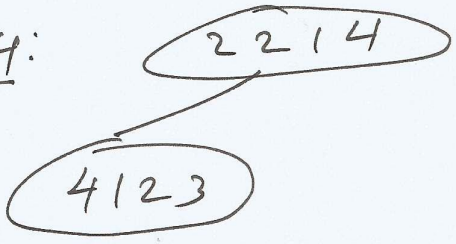
BST



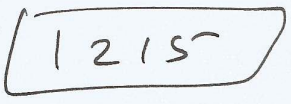
Processor

The task 3 will run for 1 unit of time. At T=4, the task 3 will finish. Now delete the highest priority task from BST and allocate it on the processor:

T=4:



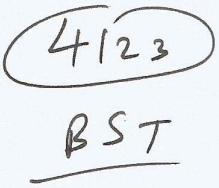
BST

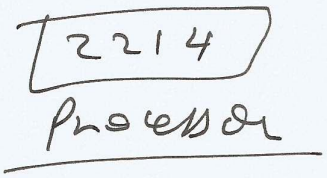


Processor

The task 1 will run for 1 unit of time. At T=5, the task 1 will finish. Now delete the highest priority task from BST and allocate it on the processor:

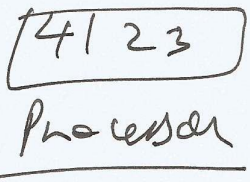
T=5 :





The task 2 will run for 1 unit of time. At T=6, the task 2 will finish. Now delete the highest priority task from BST and allocate it on the processor:

T=6 :



The task 4 will run for 2 units of time. At T=8, all tasks are finished and the BST is empty.

Sample Output : 0 0 1 3 1 2 4 4