Name: Madhusmita Oke
ID : 2018H1030194G

Description about the Source Code:
How to use custom scheduler file?

- The .sc file is placed in directory with cheddar.exe and scheduler type is selected as ‘Pipeline User Defined Protocol’ along with mentioning scheduler file (while defining core). When the task set simulation is viewed ,custom file is considered for scheduling mechanism.

Working of Code:
As the scheduler is preemptive, the priority section computes the task for the next execution after every unit time quantum. start section –initializes variables and election section – returns the task id of task to be executed.
<u>Priority section working-</u>

- Sum of remaining time of execution is calculated.
- Percentage contribution of a task to the total is given by : (execution time remaining /sum) *100
  Execution time remaining : tasks.rest_of_capacity
- Lottery tickets for each task are calculated as: 100 - percentage contribution (In this way the task contributing less to the utilization will get more number of tickets)
- Total Lottery tickets : sum of tickets assigned to each task.(sum)
- Range of lottery tickets is assigned to each task (low limit and higher limit for each task is stored based on number of tickets)
- Random number is generated using uniform (a,b,c) function within range 1 to sum of tickets
- A loop makes sure random number considered in further calculation is same as random lottery ticket number selected (As random data type variable changes value every time its accessed in the code). The loop increments ‘ticket’ value till random number becomes equal to the ‘ticket’. Then ‘ticket’ integer variable is used in further task selection.
- The task with ticket number falling in it’s range is selected for next execution (and returned in election section).

Algorithm: (for allotment of lottery tickets to each task)

```
for i tasks_range loop
    sum = sum + tasks.rest_of_capacity(i)
end loop

for i in tasks_range loop
    lottery_tickets(i) = 100 – (tasks.rest_of_capacity*100 / sum )
end loop
```

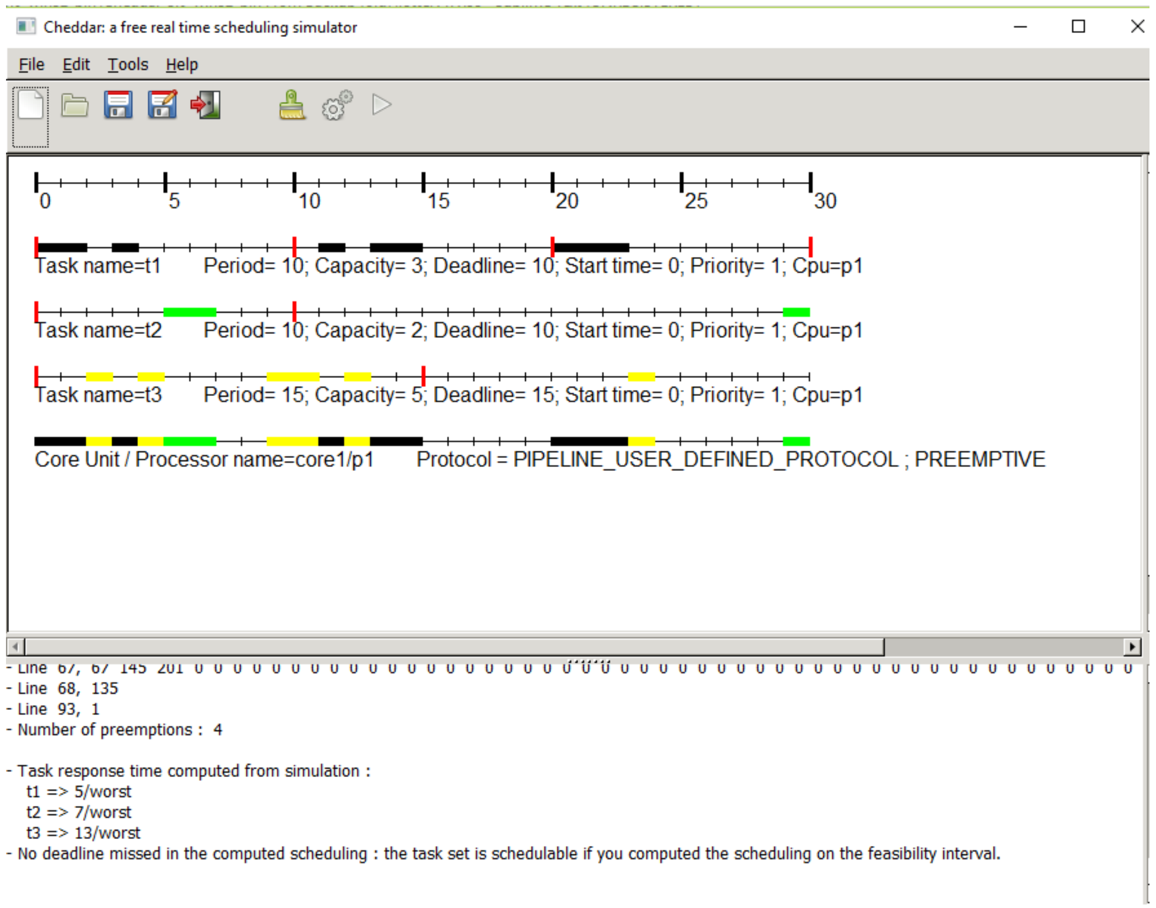
// tasks.rest_of_capacity gives the remaining execution time for each task
// lottery_tickets(i) gives tickets assigned to each i^{th} task . It is 100 – (percentage contribution to total capacity i.e. execution time of all tasks)
Thus, the number of tickets is not constant , it changes after every unit time quantum . The range of random ticket selection varies accordingly.
This ensures that task contributing less to the utilization will get more number of tickets.

Task Set examples and Simulation Screenshots:

The simulation examples given below show then timeline of execution of tasks based on selection of lottery ticket. If K ticket is selected the task which is assigned Kth ticket is selected for execution. So the tasks with more number of tickets allotted have more probability of getting chance of execution (more contribution to processor = less number of tickets as per algorithm)

Example 1:

Task	**Start Time**	**Deadline**	**Period**	**Execution Time**
T1	0	10	10	3
T2	0	10	10	2
T3	0	15	15	5



Task name=t1 Period= 10; Capacity= 3; Deadline= 10; Start time= 0; Priority= 1; Cpu=p1

Task name=t2 Period= 10; Capacity= 2; Deadline= 10; Start time= 0; Priority= 1; Cpu=p1

Task name=t3 Period= 15; Capacity= 5; Deadline= 15; Start time= 0; Priority= 1; Cpu=p1

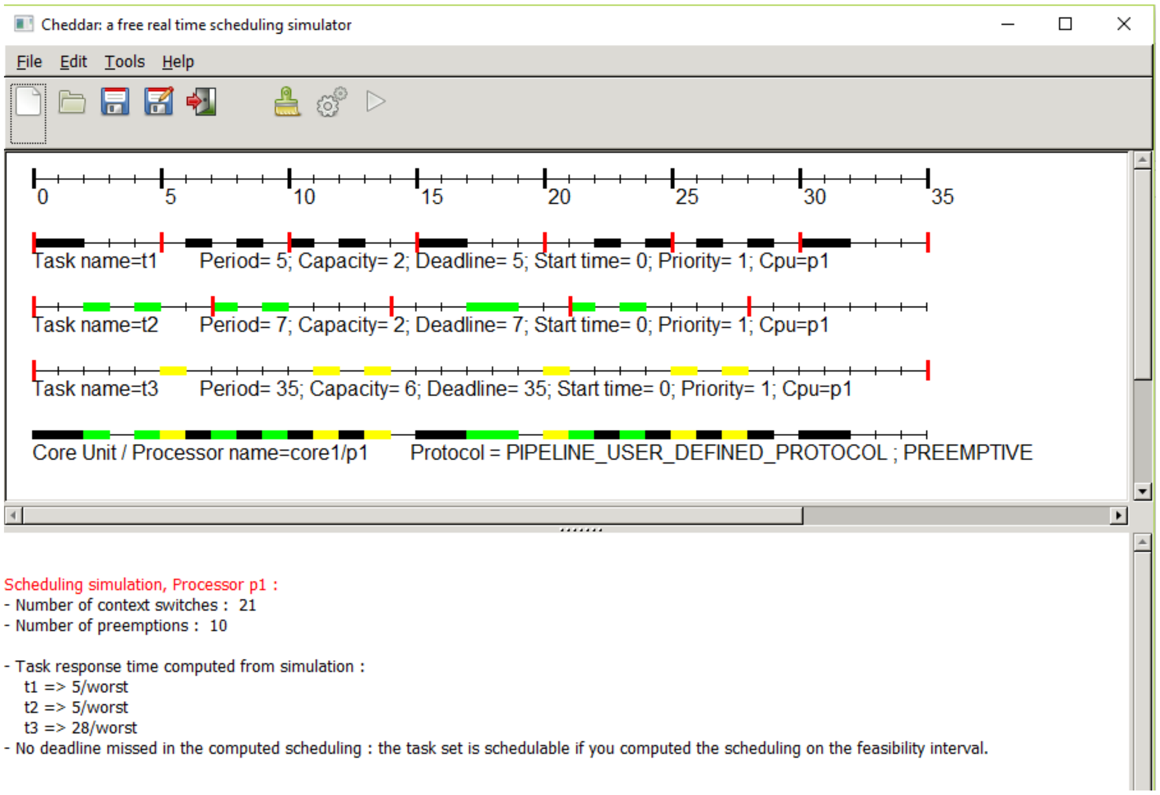
Core Unit / Processor name=core1/p1 Protocol = PIPELINE_USER_DEFINED_PROTOCOL ; PREEMPTIVE

```
- Line  67,  67  145  201  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
- Line  68,  135
- Line  93,  1
- Number of preemptions :  4

- Task response time computed from simulation :
   t1 => 5/worst
   t2 => 7/worst
   t3 => 13/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

Example 2:

Task	Start Time	Deadline	Period	Execution Time
T1	0	5	5	1
T2	0	7	7	2
T3	0	35	35	5



Task name=t1 Period= 5; Capacity= 2; Deadline= 5; Start time= 0; Priority= 1; Cpu=p1

Task name=t2 Period= 7; Capacity= 2; Deadline= 7; Start time= 0; Priority= 1; Cpu=p1

Task name=t3 Period= 35; Capacity= 6; Deadline= 35; Start time= 0; Priority= 1; Cpu=p1

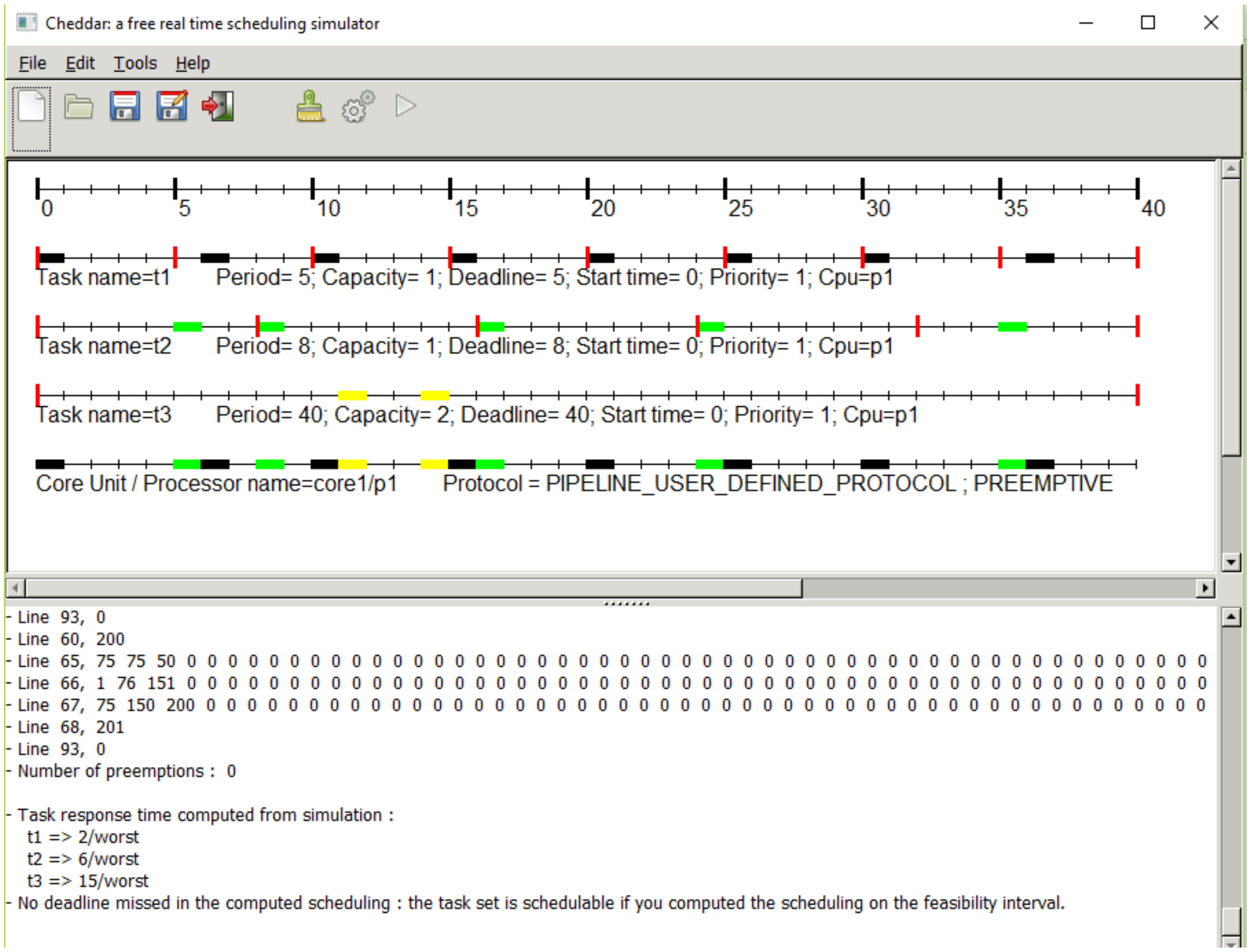
Core Unit / Processor name=core1/p1 Protocol = PIPELINE_USER_DEFINED_PROTOCOL ; PREEMPTIVE

```
Scheduling simulation, Processor p1 :
- Number of context switches :  21
- Number of preemptions :  10

- Task response time computed from simulation :
    t1 => 5/worst
    t2 => 5/worst
    t3 => 28/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```

Example 3:

Task	**Start Time**	**Deadline**	**Period**	**Execution Time**
T1	0	5	5	1
T2	0	8	8	1
T3	0	40	40	2



Task name=t1 Period= 5; Capacity= 1; Deadline= 5; Start time= 0; Priority= 1; Cpu=p1

Task name=t2 Period= 8; Capacity= 1; Deadline= 8; Start time= 0; Priority= 1; Cpu=p1

Task name=t3 Period= 40; Capacity= 2; Deadline= 40; Start time= 0; Priority= 1; Cpu=p1

Core Unit / Processor name=core1/p1 Protocol = PIPELINE_USER_DEFINED_PROTOCOL ; PREEMPTIVE

```
- Line 93, 0
- Line 60, 200
- Line 65, 75 75 50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- Line 66, 1 76 151 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- Line 67, 75 150 200 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
- Line 68, 201
- Line 93, 0
- Number of preemptions : 0

- Task response time computed from simulation :
   t1 => 2/worst
   t2 => 6/worst
   t3 => 15/worst
- No deadline missed in the computed scheduling : the task set is schedulable if you computed the scheduling on the feasibility interval.
```