**CompE-571**

Homework #3

Sushma Kuchipudi-820850180

Krishna Dogney-820891026

Programming language: Python

IDLE: 2.7 version

Data Structures Used:

* Dictionaries: To store task parameters like period, execution times at different frequencies etc. in the form of keys and values.
* Lists: To maintain a queue of tasks to be executed by CPU like a waiting list and also to store the order of tasks to be executed

Command in cmd: folder>program\_name textfile\_name (C:\Users\sushm\Desktop>EDF.py input2.txt)

The parameters of each task i: e each line in the text file is first read into dictionaries(tasks) so we can sort as per key and its value later during scheduling. example: {'name': 'w1', 'period': 520, 'wcet@384': 141}

EDF Algorithm:

* First, the EDF conditionK:\multimedia comm\Screenshot004.jpg<=1 for the given set of tasks is checked using the execution times at maximum frequency. If the condition is being satisfied, we proceeded with the rest of the scheduling algorithm which is written in the form of function EDF ().
* EDF():At t=0, we appended all the tasks in the list called lists and from t=1 it is constantly being checked if any new task has arrived or not.
* Every time a task has arrived its remaining execution time to be updated to its original execution time and the current deadline of the task to be updated to its next corresponding deadline i:e the time at which it next arrives.
* As time parameter i.e t increases the deadline of every task is decremented by 1. This is like notifying the scheduler that deadlines are approaching.
* If the list is not empty i.e the CPU has tasks in its queue that are to be scheduled and executed, then the tasks in the list are sorted as per the key-‘deadline’ or else the CPU state is set to be IDLE.
* After sorting the queue i.e lists, the one with the least deadline will be executed at that time t and this sorting is done for every t to check for pre-emption and to make sure that no task is missing its corresponding deadline.
* We maintained a list (with name a) with time as index and task that has been executed at that time t as an item in that list. (ex:a=[w1,w1,w1,w2,w2,………]).This is to calculate the start and end times later for each task till 1000 sec later on.
* Every time a task being executed at time t its remaining execution time is decremented by 1 and then after decrementing if remaining execution time (i:e remexetym variable in our program) is equal to zero then that particular task will be removed from the queue.
* After calling this EDF function from t=0 to 1000 the output will be the list with time as index and the task running at that time as an item in that list.
* So, then loop is iterated from 0 to 1000 and we checked if task at t and t+1 are same and if they are not, then a task has ended and new task has started. We updated end variable for current task and then displayed the asked values start time, task, end time, frequency, energy used for each task and then updated the start value for next task.
* We also found the idle percentage and the time for which each task has run in the span of 1000 sec and total energy cost of the schedule.

RM Algorithm:

* First, the EDF condition K:\multimedia comm\Screenshot.jpg for the given set of tasks is checked using the execution times at maximum frequency. If the condition is being satisfied, we proceeded with the rest of the scheduling algorithm which is written in the form of function RM ().
* RM ():At t=0, we appended all the tasks in the list called lists and from t=1 it is constantly being checked if any new task has arrived or not.
* Every time a task has arrived its remaining execution time to be updated to its original execution time.
* If the list is not empty i.e the CPU has tasks in its queue that are to be scheduled and executed, then the tasks in the list are sorted as per the key-‘period’ this time or else the CPU state is set to be IDLE.
* After sorting the queue i.e lists, the one with the least period will be executed at that time t and this sorting is done for every t to check for pre-emption i.e to check if a task with lesser period is in the queue.
* We maintained a list (with name a) with time as index and task that has been executed at that time t as an item in that list. (ex:a=[w1,w1,w1,w2,w2,………]).This is to calculate the start and end times later for each task till 1000 sec later on.
* Every time a task being executed at time t its remaining execution time is decremented by 1 and then after decrementing if remaining execution time (i.e remexetym variable in our program) is equal to zero then that particular task will be removed from the queue.
* After calling this RM function from t=0 to 1000 the output will be the list with time as index and the task running at that time as an item in that list.
* The start and end times has been calculated in the similar manner as mentioned in the EDF algorithm.
* We also found the idle percentage and the time for which each task has run in the span of 1000 sec and total energy cost of the schedule.

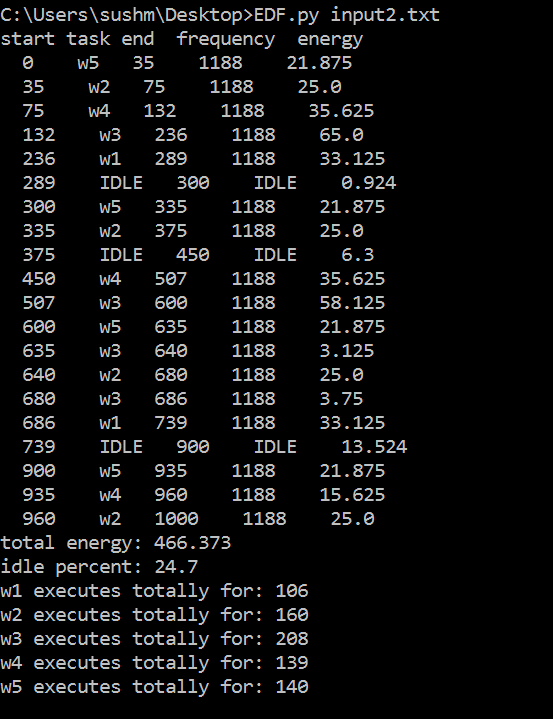
EE EDF Algorithm:

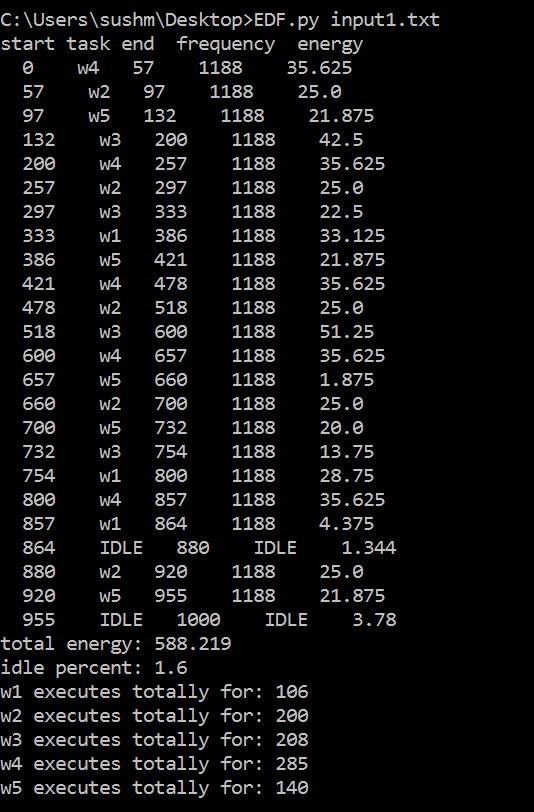
* All the combination of tasks with different execution times at different frequencies that satisfies the EDF condition have been found and the set of tasks that run at different frequencies that can be executed with least total energy consumed has been considered for scheduling.
* So, we found out at which frequency a task runs every time it arrives. (Ex: w1 runs at 918Ghz, w2 at 1188,w3 at 918,w4 at 1188,w5 at 1188 every time they arrive.)
* Then the execution times and frequencies have been updated into the dictionary tasks accordingly and the above EDF algorithm has been used to schedule the tasks for every t till 1000 sec.
* The start and end times are calculated as described above.

EE RM Algorithm:

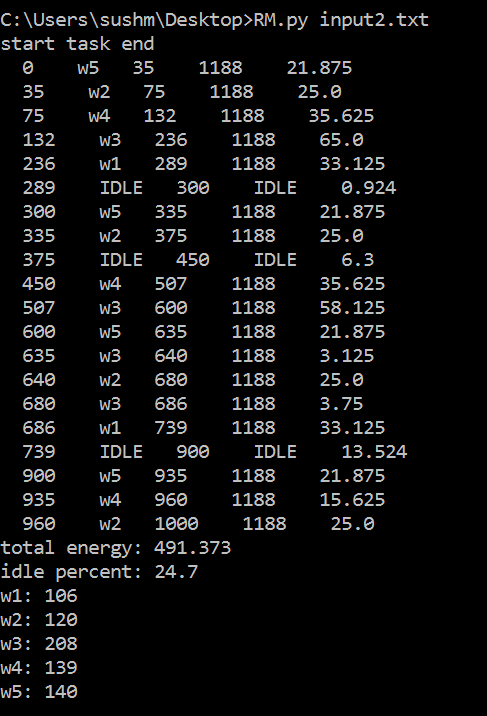
* All the combination of tasks with different execution times at different frequencies that satisfies the RM condition have been found and the set of tasks that run at different frequencies that can be executed with least total energy consumed has been considered for scheduling.
* So, we found out at which frequency a task runs every time it arrives. (Ex: w1 runs at 918Ghz, w2 at 1188, w3 at 918,w4 at 1188,w5 at 1188 every time they arrive.)
* Then the execution times and frequencies have been updated into the dictionary tasks accordingly and the above RM algorithm has been used to schedule the tasks for every t till 1000 sec.
* The start and end times are calculated as described above.

Results:

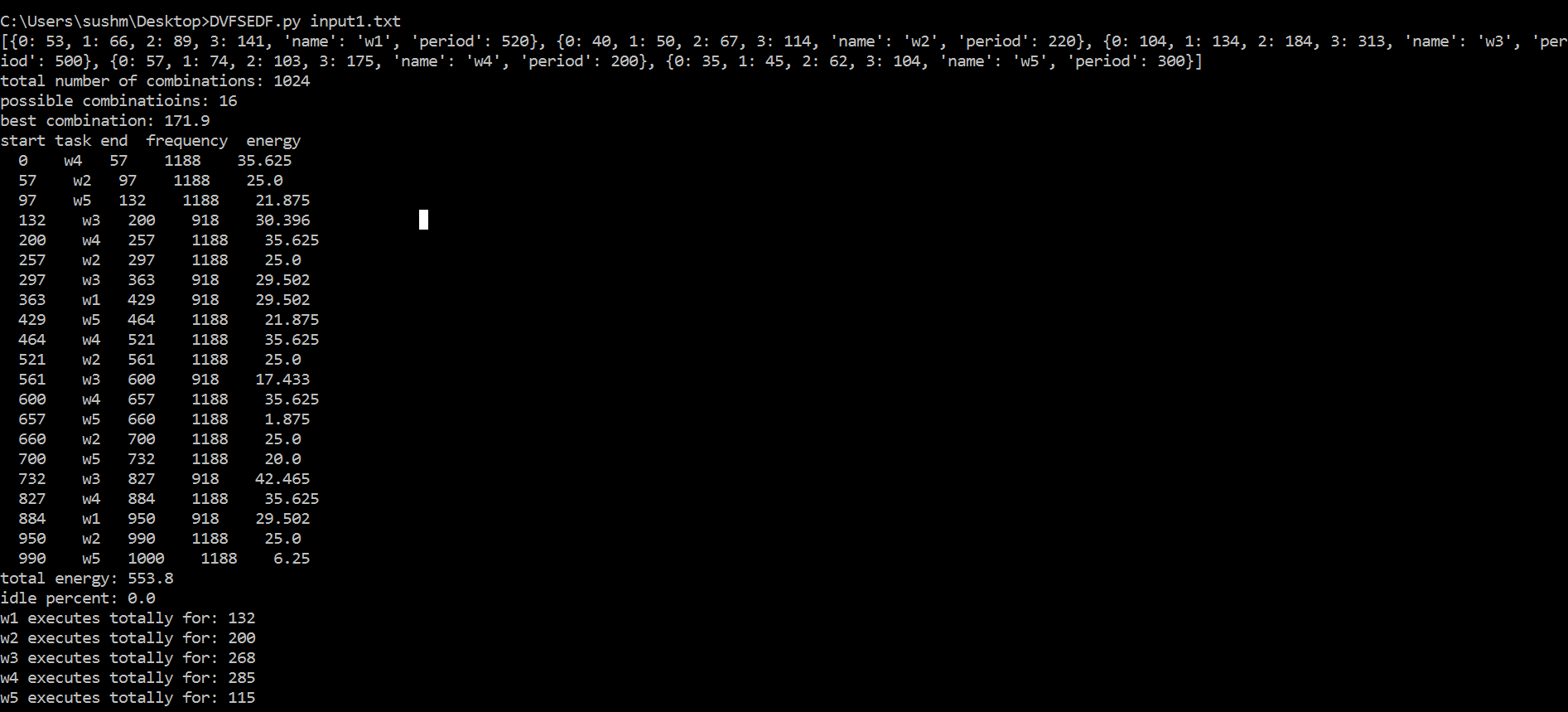
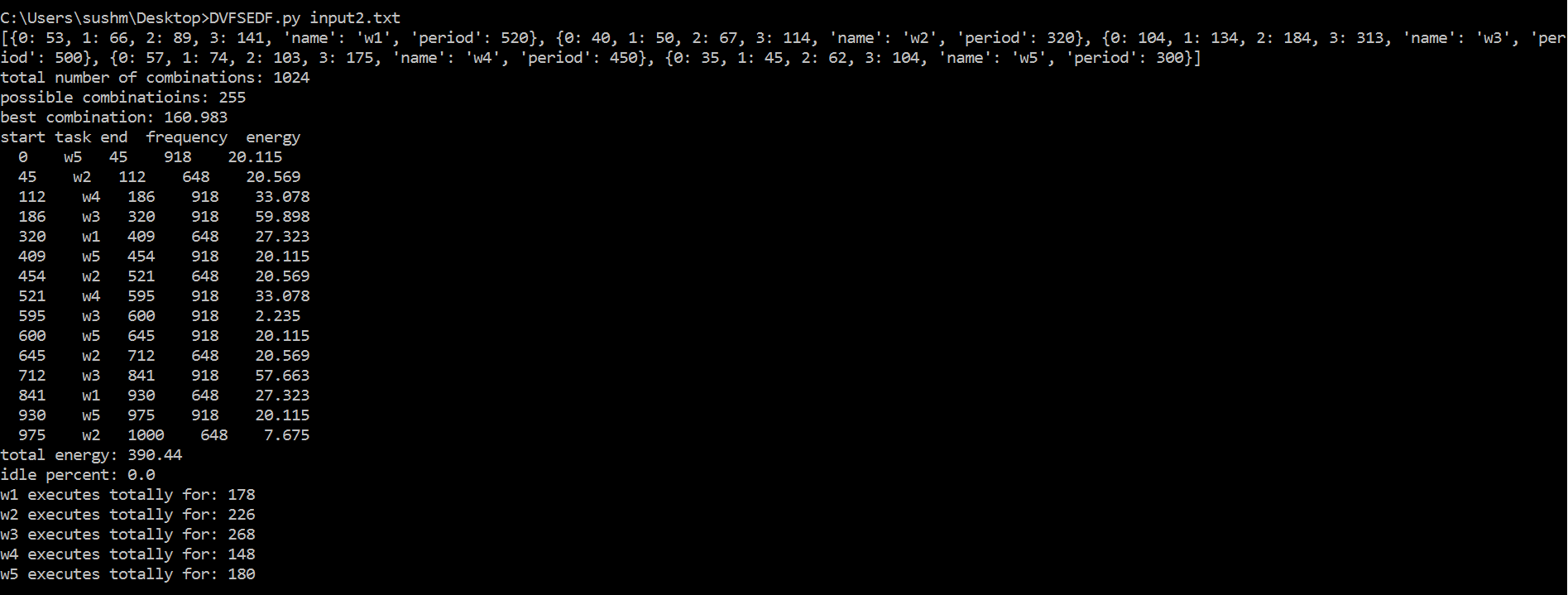
EDF for input1.txt: EDF for input2.txt:



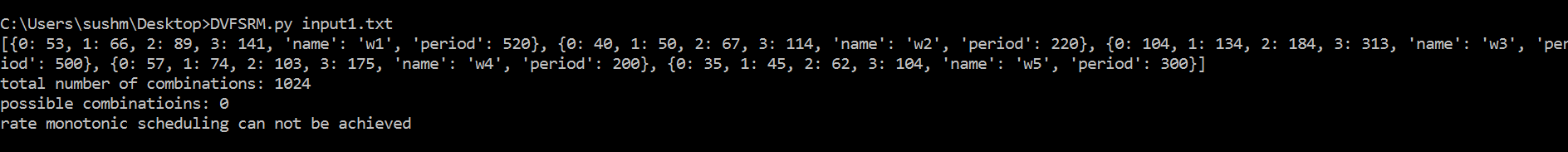
RM for input1.txt: RM for input2.txt:



EE EDF for input1.txt: EE EDF for input2.txt:



EE RM for input1.txt:



EE RM for input2.txt:

