Assignment 2

In this assignment, you will develop a tool that simulates an EDF scheduler. Your script will take in a file that contains data about the task system and will output data about when specific jobs run into a file. Additionally, you will modify your visualizer to handle periodic tasks.

Part 1: Setting up your workspace.

1. In your home directory on the server, make a new directory called HW2.

```
mkdir HW2
```

2. Copy the sample data to your directory:

```
cd HW2
cp /home/shared/executions.json .
cp /home/shared/tasks.json .
```

3. Copy the template to your directory and rename it `EDF.py`:

```
cp /home/shared/template2.py EDF.py
```

4. Copy your visualization tool from the first assignment to the new directory:

```
cp ../HW1/visualizer.py .
```

Part 2: The assignment.

A - Modifying the visualization tool

1. Make sure arrows are visible even when a task misses its deadline. The easiest way to do this is to make sure that the rectangles of execution are drawn before the up- and down-arrows.
2. Change the script to handle periodic tasks. If a task has a period that is non-zero, draw the appropriate job release arrows and deadline arrows until the time `max_t`. (Note that the tasks now also have a "period" field.)
3. Test your modified tool by using the supplied executions and looking at the pdf.

```
python visualizer.py executions.json
ps2pdf schedule.ps partA.pdf
pdfcrop partA.pdf partA.pdf
```

B - Producing executions

1. Check that `EDF.py` takes the `max_t` from the input file and writes it to the output file. The input file is the first argument, and the output file is the second argument.

```
      python EDF.py tasks.json new_executions.json
```

2. Modify `EDF.py` to write the task set information to the output file.
3. Toward the top of the file `EDF.py`, fill in the function that adds an execution to the output file. Recall that each execution includes the components `task number, job number, start time,` and `end time`.
4. Add executions to the output file based on how EDF would schedule the task system from time 0 to `max_t`.
5. Test your EDF simulator with `tasks.json`. Produce `new_executions.json` as the output. Then look at the schedule produced by running it through your visualizer, and verify that it is indeed following the rules of EDF. Name this pdf `PartB.pdf`.

```
      python EDF.py tasks.json new_executions.json
      python visualizer.py new_executions.json
      ps2pdf schedule.ps PartB.pdf
      pdfcrop PartB.pdf PartB.pdf
```

Part 3: Submitting your assignment.

Your assignment should be in `/home/<yourCSlogin>/HW2`. I will collect homework from here and check the last time that each file was modified. This assignment is due on Sept. 17, 2018 at 9:05am EST. If the files have been edited after that time, I will assume that you have chosen to use one or more late days. If you would like to continue tweaking your solution, do so in a different folder. Make sure that the following files are in `/home/<yourCSlogin>/HW2`, especially if you were working in a different directory or on a different machine:

- `PartA.pdf`
- `visualizer.py`
- `PartB.pdf`
- `new_executions.json`
- `EDF.py`

Your schedule visualizer and execution generator will also be tested with additional inputs.