CS 162A - Operating Systems
Project 2

Disk Scheduling

For this project, you are going to simulate servicing disk requests. You will need to implement four disk scheduling algorithms:

1. First Come First Served (FCFS)
2. Shortest Seek Time First (SSTF)
3. SCAN
4. LOOK

Disk Scheduling Algorithms are all **non-preemptive**. That means if a request is being serviced and a new request comes in it will be ignored until the current request is being done. Each Disk Scheduler will have a corresponding class that implements the DiskScheduler interface. The **DiskScheduler** interface will have the following methods:

1. **public void addRequest(int arrivalTime, int cylinderAddress)** - this method adds a request to be scheduled by the scheduler.
2. **public DiskSchedule computeSchedule()** - this method takes in all requests added through addRequests(), schedules the requests according to the type of algorithm, and computes the Tang chart.

You will need to create the following classes, which should implement DiskScheduler:

1. **FcfsScheduler.java** - class the implements the FCFS disk scheduling algorithm.
2. **SstfScheduler.java** - class that implements the SSTF disk scheduling algorithm.
3. **ScanScheduler.java** - class that implements the SCAN disk scheduling algorithm.
4. **LookScheduler.java** - class that implements the LOOK disk scheduling algorithm.

In order to create schedulers, you will need to provide a SchedulerFactory class which has one method:

- **public static DiskScheduler createScheduler(String code, int headAddress)** - this should create a disk scheduler object based on the supplied code, assuming that the head of the disk is in the cylinder address specified by the headAddress argument. The codes are:
    - FCFS - for the FcfsScheduler
    - SSTF - for the SstfScheduler
    - SCAN - for the ScanScheduler
    - LOOK - for the LookScheduler

  Note that each code use **all uppercase letters.**

A sample implementation of the SchedulerFactory will be provided. You can modify it as you wish, as long as you do not change the method signature of **createScheduler()**.

The DiskSchedule object that is returned by computeSchedule() will contain information for the Tang chart. It should have the following methods:

1. **public List<ServicedRequest> getRequestOrder()** - returns the list of serviced requests in the order they were done.
2. **public int averageResponseTime()** - returns the average response time. The response time for a requst is computed as:
   <time when request was serviced> - <arrival time of request>

The ServicedRequest class is used to contain information about a serviced request. It should should contain the following methods:

1. **public int getRequestID()** - returns the ID of the request. The ID of the request depends on when it is added to the scheduler. The first request added will have an id of 0, the second will have an id of 1, and so on.
2. **public int timeServiced()** - returns the time that the request was serviced.

For all algorithms, assume the following:
1. Assume that there is zero rotational latency and zero read/write time. Therefore, we only consider the time it takes to get to the cylinder address.
2. Assume that it takes 1 ms to move from cylinder C to cylinder C+1 or C-1.
3. For SCAN/LOOK, assume that the head is always going towards the higher addresses.
4. If there are any ties for servicing the next request, prioritize the request with the smaller request ID

You will also need to create an application(**ScheduleDisks.java**) that reads in service requests from a file, **io-requests.txt** and will output the schedule into another file, **disk-schedules.txt**. This should use the DiskScheduler objects mentioned above.

The **io-requests.txt** will have 1 or more sets of requests to schedule. Each set will consist of 3 or more lines of text. Each set will have the following format:
- The first line of each set will contain a string <code>, and an integer <head>, separated by spaces.
  - The **<code>** specifies the **type** of scheduling algorithm to use. This will be FCFS, SSTF, SCAN, LOOK (note: all uppercase letters). This corresponds to the code for the scheduling algorithm.
  - The **<head>** provided is the **starting cylinder address** of the head. This will be used by the scheduling algorithm.
- The second line of each set will contain one integer <n> . This integer specifies the

number of requests to be serviced.

- The next **<n>** lines will contain the actual requests. Each request will consist of two integers separated by spaces, the <arrivalTime> and the <cylinderAddress>
    - The <arrivalTime> is the actual time that the request arrived in the system. Assume that the system starts at time 0.
    - The <cylinderAddress> is the address of the request. Assume that maximum possible cylinder address to be **2047.**

The sets of requests will be terminated by the line:

```
END 0
```

Here is an example of how the io-requests.txt file may look like:

```
FCFS 0
3
0 100
0 200
0 300
SSTF 150
4
0 160
0 140
0 115
30 130
END 0
```

For each set of requests, your application should print two lines in the **disk-schedules.txt** file:

- The first line should contain the requests serviced. For each request, you should print the **request ID** and the **time that the request is serviced** that request. This should be printed in the order the requests were serviced.
- The second line should contain one number: the **average response time** over all requests. It should be formatted to 2 decimal places.

Here is an example of how the disk-schedules.txt file will look like:

```
R0 100 R1 200 R3 300
200.00
R0 10 R1 30 R4 40 R3 55
33.75
```

Bonus(10 pts): Implement CLOOK and CSCAN algorithms. You will need to create the following classes:

1. CLookScheduler - implements the CLOOK algorithm.

2. CScanScheduler - implements the CSCAN algorithm.

These classes should, of course be added to the application, and the SchedulerFactory class for their corresponding codes.