BLG 335E ANALYSIS OF ALGORITHMS

HW#2

Due Date: 12 December 2019, 23:59

Homework Policy

- Use comments whenever necessary to explain your code. Also, **write your name on top of each file you are sending in comment!**
- Do not forget to handle exceptions and print error messages!
- Your code must be written in C++, and should be able to be compiled and run using g++ compiler. ITU servers provide g++ compiler, you can test your program by connecting to the servers using ssh (if you don't know how to do it, click here).
- Your program must take the name of the file that contains events (**see homework description for events file format**) from the **command line**. Therefore, it must be executed as follows:

  ./your_executable_file events_file_name

- Your program must produce **output exactly same as defined in the homework description below**.
- IMPORTANT: This is an individual assignment! You are expected to act according to Student Code of Conduct, which forbids all ways of cheating and plagiarism. It is okay to discuss the homework with others, but it is strictly forbidden to use all or parts of code from other students' codes or online sources and let others do all or part of your homework. **Your codes will be checked using plagiarism detection software. In case of cheating occurs, you will be subject to disciplinary actions.**
- If you have any questions, write it to **HW1 message board** on Ninova.

Problem Description

In this homework, you are asked to implement an event-scheduler as a MIN-HEAP. A file keeps a list of events with the following format:

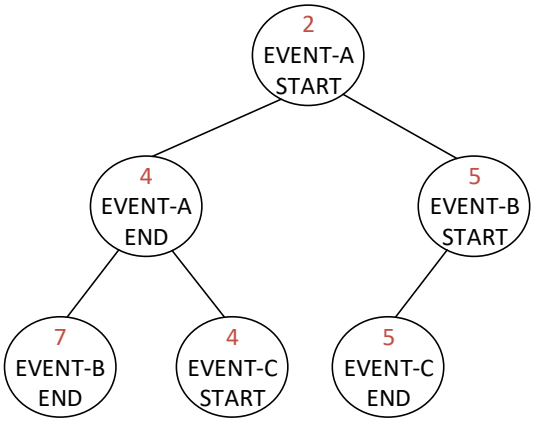
EVENT-NAME START-TIME END-TIME

where EVENT-NAME is the name of the event, START-TIME is the scheduling time of the event and END-TIME is the finishing time of the event. Following is an example **events.txt** file which consists of three events: EVENT-A which starts at time 2 and ends at time 4, EVENT-B which starts at time 5 and ends at time 7, and EVENT-C which starts at time 4 and ends at time 5.

events.txt

```
EVENT-A     2     4
EVENT-B     5     7
EVENT-C     4     5
```

The event scheduler starts by reading the list of events from an event file **whose name is supplied as a command line parameter, as mentioned in homework policy section**. Event scheduler creates an event MIN-HEAP using the firing times of the events as keys. Note that, each listed event in the events file produces two events: (1) an event at the start time and (2) an event at the end time. Therefore, the heap nodes must store event type, either start or end, as well as the event name. An event MIN-HEAP for the given example file is as follows:



There is a virtual clock in the system which starts from 0 and ticks by 1 time unit. At each clock tick, event scheduler checks the top event in MIN-HEAP to see whether there is a scheduled event at that time or not:

- If there is no scheduled event for that time, the program prints out:

  TIME **T**: NO EVENT

  where **T** is the current time.

- If there is a START event for the scheduled time, then the program prints out:

  TIME **T**: **EVENT-NAME** STARTED

  where **T** is the current time and **EVENT-NAME** is the name of the event. Afterwards, the event is removed from the event MIN-HEAP.

- If there is an END event for the scheduled time, then the program prints out:

  TIME **T**: **EVENT-NAME** ENDED

  where **T** is the current time and **EVENT-NAME** is the name of the event. Afterwards, the event is removed from the event MIN-HEAP.

Please note that, there may be more than one events scheduled for the same time (e.g., both EVENT-A END and EVENT-C START events are scheduled for time 4 in the example above). Therefore, you have to check heap till the scheduling time on top of the event MIN-HEAP becomes larger than the current time.

Program Run and Output for The Given Example

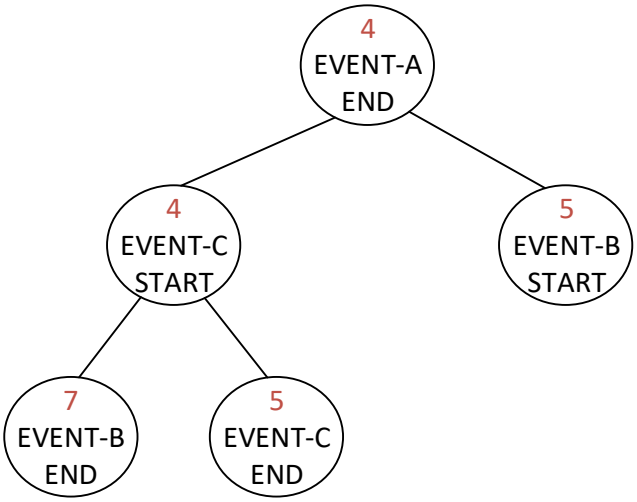
At time T = 1, there is no scheduled event. Therefore, the program prints out:

TIME 1: NO EVENT

The clock ticks and at time T=2, there is a scheduled START event on the top of the event MIN-HEAP. Therefore, the program prints out:

TIME 2: EVENT-A STARTED

Then, the node is removed from the heap and heap becomes:



The top element has a scheduled time 4, which is larger than the current time T=2. Therefore, the clock ticks to the next time, i.e., T=3.

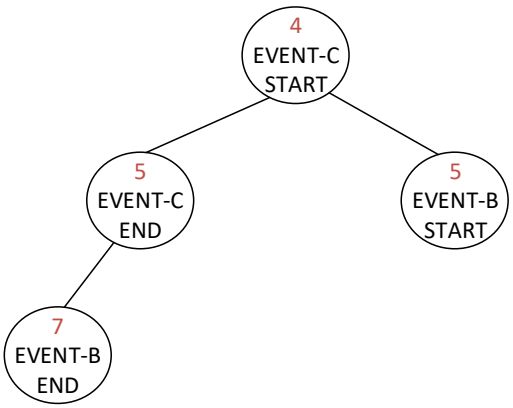
At time T=3, there is no scheduled event. Therefore, the program prints out:

TIME 3: NO EVENT

The clock ticks and at time T=4, there is a scheduled END event on the top of the MIN-HEAP. Therefore, the program prints out:

TIME 4: EVENT-A ENDED

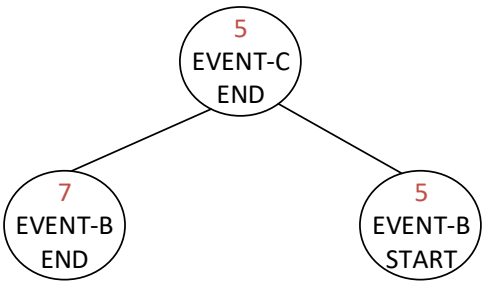
Then, the node is removed from the heap and heap becomes:



The top element has a scheduled time 4, which is equal to the current time T=4. Therefore, the program prints out:

TIME 4: EVENT-C STARTED

Then, the node is removed from the heap and heap becomes:

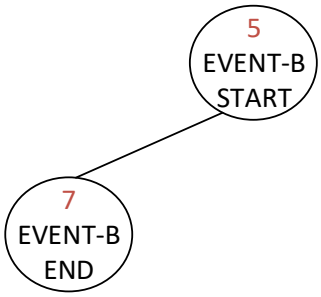


The top element has a scheduled time 5, which is larger than the current time T=4. Therefore, the clock ticks to the next time, i.e., T=5.

At time T=5, the top element has a scheduled time 5, which is equal to the current time. Therefore, the program prints out:

TIME 5: EVENT-C ENDED

Then, the node is removed from the heap and heap becomes:



The top element has a scheduled time 5, which is equal to the current time T=5. Therefore, the program prints out:

TIME 5: EVENT-B STARTED

Then, the node is removed from the heap and heap becomes:

The top element has a scheduled time 7, which is larger than the current time T=5. Therefore, the clock ticks to the next time, i.e., T=6.

At time T=6, there is no scheduled event. Therefore, the program prints out:

TIME 6: NO EVENT

The clock ticks and at time T=7, there is a scheduled END event on the top of the MIN-HEAP. Therefore, the program prints out:

TIME 7: EVENT-B ENDED

Then, the node is removed from the heap and heap becomes empty.

Since the MIN-HEAP is empty, all the events are scheduled. The program prints out:

TIME 7: NO MORE EVENTS, SCHEDULER EXITS

Afterwards, the program exits gracefully since there is no more events and the user is informed!

Submission

Submit your homework files through Ninova. Please zip and upload all your files. You are going to submit the following files:

(a) All your .h (if any) and .cpp files

(b) A .pdf file as your report. Your report should be clear and detailed. Otherwise, it may result in a grade loss for you.

(c) A .txt file explaining how to compile and run your code (in a 1 or 2 line)

Use your student ID as file names. That is, **your_student_id.zip** for the .zip file, **your_student_id.cpp** for the main C++ source file and **your_student_id.pdf** for the report.