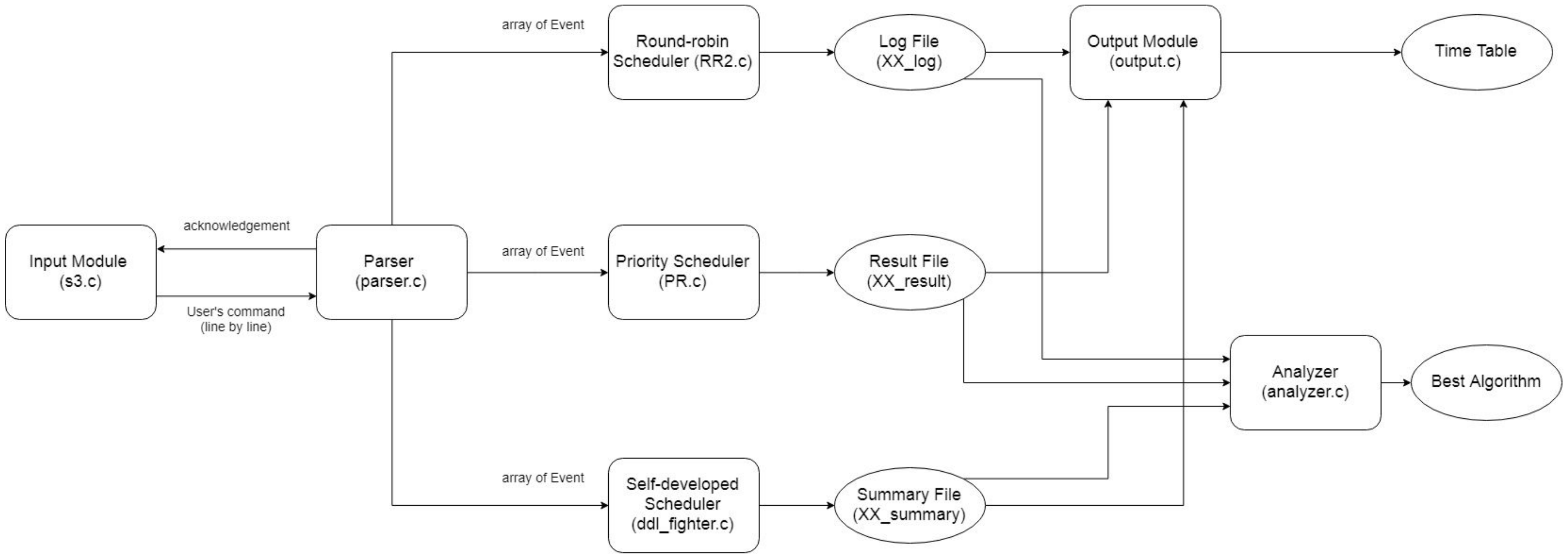
Group 01

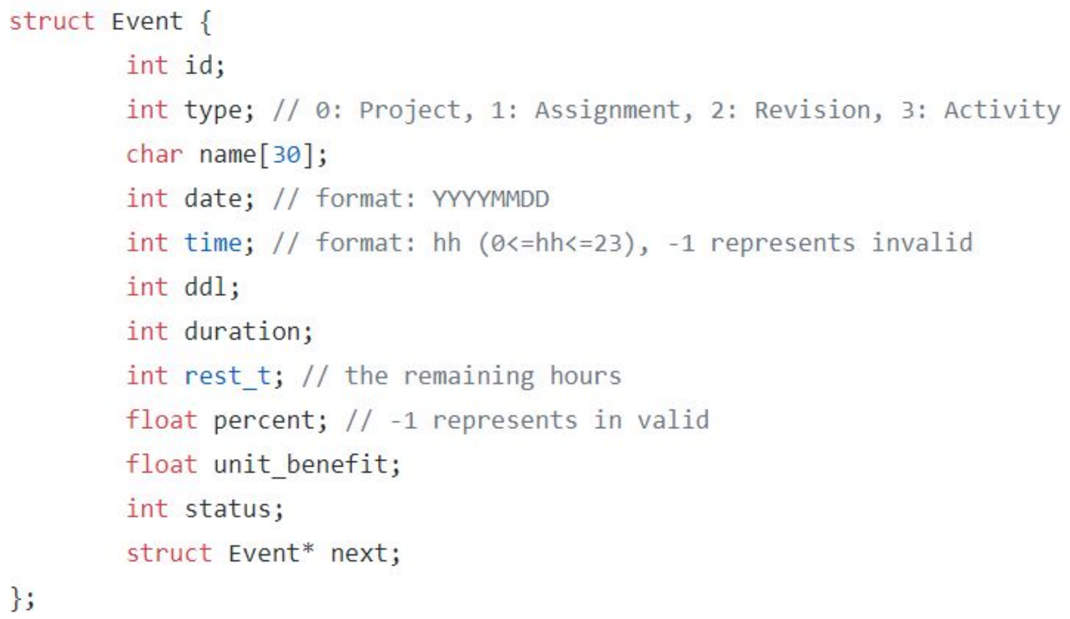
LIU Fengming 15104126D
LIU Le 15103617D
YU Jing 16098537D
DING Dashan 17082316D

Program Structure



Data Structure

All events are stored as a structure called Event



```
struct Event {
        int id;
        int type; // 0: Project, 1: Assignment, 2: Revision, 3: Activity
        char name[30];
        int date; // format: YYYYMMDD
        int time; // format: hh (0<=hh<=23), -1 represents invalid
        int ddl;
        int duration;
        int rest_t; // the remaining hours
        float percent; // -1 represents in valid
        float unit_benefit;
        int status;
        struct Event* next;
};
```

Scheduling -- Priority

Data Structure:

a linked list of struct Events

Algorithm:

(1) Sort the linked list in the order of Priority.
    (a) First, we sort the linked list in the order of Project -> Assignment -> Revision -> Activity so that the scheduler only need to process the linked list in natural order.

Scheduling -- Priority

Algorithm:

(2) The program traverse the linked list.

(a) If the current event is a project or an assignment, then it allocates the time to the current event until it is finished. If it cannot be finished, then we only finish a part of it until the time reaches deadline or the end of the time period.

(b) If the current event is a revision or an activity, then it checks if the time period of the revision or activity is available. (An array is used to store the information of time slots so we know if it is occupied) If it is available, then we allocate the time to it; or else, we reject the event.

Scheduling -- Round Robin

Data structure:

a linked list of structure Event

Algorithm:

(1) If the head of the linked list is a Project / Assignment, allocate a quantum to the Event first and progress the clock.
    (a) If the clock exceeds the deadline of the Event or the Event has been finished, back tune the clock to the proper time, dequeue the Event and continue
    (b) Else, insert the Event to the tail, move the head to the next and continue

Scheduling -- Round Robin

Algorithm:

(2) If the head of the linked list is a Revision / Activity

(a) If the clock exceeds the right time, reject the Event and continue
(b) If the the right time is in the future, insert the Event to tail and continue
(c) If the clock is at the right time, but the duration exceeds the remaining time of the day, reject the Event and continue
(d) If the clock is at the right time and the remaining time is enough for the duration, accept and allocate the Event, dequeue the Event and continue

Scheduling -- Round Robin

Algorithm:

(3) When the clock reaches the end of the period, reject all the remaining Events

Scheduling -- The Deadline Fighter Algorithm

- Scoring System
- Objective
  - First off, grab MAXIMUM score from projects/assignments
  - Then, schedule as many revisions as possible
  - Finally, schedule as many activities as possible
- Mechanism
  - Mathematically provable MAX score from projects/assignments
  - Greedy algorithm for revisions/activities. Does NOT guarantee max for revisions (the Knapsack Problem, NP-complete), but close enough in this application.