ELC303-B 2021
Assignment: Meeting Planner

May 19, 2021

1 Objective

A meeting planner is important to remember one's meetings and keep them organized. This project aims to implement a simple version of a meeting planner using a binary search tree.

2 Submission Instruction

You are expected to submit using the online submission system using the upload file(s) link.

- All source files (.cpp) and header files (.h) used should be uploaded to the blackboard classroom. Do **NOT** submit the whole project or zip files.
  If you submit the whole IDE(eclipse) project or any compressed files of any kind, they will not be considered in evaluation
- Make sure your program reads from an input text file and prints its outputs on screen. **The names of the input file will be provided as input when running the program**.
- Your output will be **AUTOMATICALLY** compared with the expected output for each test case, so make sure your output **is exactly the same** as the output provided for each case in the example test cases.

Submission Deadline is June 10, 2021 @ 11PM.

- Submit even if your code is partially working. **Late submissions are not allowed**
- Write the code yourself. Plagiarism (code copying) => **-10 For Both Parties**

3 Project Overview

- This project focuses on the implementation of a meeting planner using binary search tree.

- The planner saves meetings in the form meeting title, meeting day, and meeting hour as shown in Figure 1. Meeting title is a string of letters, special characters, numbers, etc. Meeting day is an integer from 1 to 365. Meeting hour is an integer from 0 to 23.

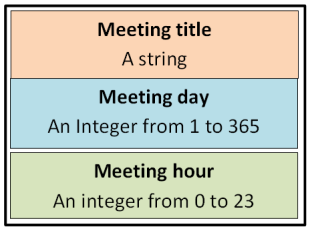

Figure 1: meeting information

- The planner should be able to
  1. add meetings with title, day, and hour,
  2. find a meeting by day and hour,
  3. modify the title of a saved meeting,
  4. delete a saved meeting,
  5. print the saved meetings sorted by day and times in an ascending order.(***You may need tree traversal here***).

4 Typical Operations

The planner should be able to do the following operations

1. **ADD** a meeting using the ADD command that takes the form

   ADD "title" day hour

   The planner saves the meeting only if no other meetings are already saved at the same day and hour (i.e. no conflict). Otherwise, it prints a conflict message with the input date **Conflict day hour**.
   For example the following command asks the planner to add a meeting with title "Lecture" on day 300 and hour 12

   ADD "Lecture" 300 12

   The planner saves the meeting only if no other meetings are saved on day 300 hour 12. Otherwise, it prints a conflict message **Conflict 300 12**

2. **Find** a meeting by its day and hour using the Find command that takes the form

   Find day hour

   The planner prints the meeting title if there is a saved meeting at the input day and hour. Otherwise it prints an empty slot message with the input day and hour **Empty day hour**

For example the following command asks the planner to find the meeting on day 300 and hour 12

Find 300 12

If there is a saved meeting on day 300 and hour 12, the planner returns the meeting title (for example, "Lecture"). If there is no saved meeting, the planner prints a message **Empty 300 12**.

3. **Modify** the title of a meeting by its day and hour using the MOD Command that takes the form

   MOD "newtitle" day hour

   The planner changes the title of the saved meeting at the input day and hour to "newtitle". If there is no meeting at the input day and hour, it return an empty slot message with the input date and hour **Empty day hour**.
   For example, the following command modifies the name of the meeting on day 300 and hour 12 to "Cinema"

   MOD "Cinema" 300 12

   If there is a saved meeting on day 300 and hour 12 (for example "Lecture"), it changes its title to "Cinema". Otherwise, it prints a message **Empty 300 12**

4. **Delete** a meeting by its day and hour using the DEL command that takes the form

   DEL day hour

   If a meeting is saved at the input day and hour, the planner deletes it. Otherwise, it prints and empty slot message with the input day and hour **Empty day hour**.
   For example, the following line asks the planner to delete the meeting on day 20 and hour 10.

   DEL 20 10

   Assuming no such meeting exists, the output should be **Empty 20 10**

5. **Print** all the meetings using the print command which takes the form

   Print

   If the planner is empty, it prints **Empty Planner**. Otherwise, it prints all the saved meetings **sorted by their day and hour (in an ascending order)**. Each meeting on a separate line with its title, day, and hour in the form

   "title of 1^{st} meeting" day hour
   "title of 2^{nd} meeting" day hour
   $\vdots$
   "title of last meeting" day hour

Note that all outputs are case sensitive
Note that all commands are case sensitive

4.1 ERROR Handling

You should check for the correctness of every input line in the following order

- If the Input file does not exist, you should print **File not Found**
- The only valid commands are ADD, Find, MOD, DEL, and Print. Any other command should print **Invalid command**
- Valid command with wrong number of input arguments should print **Invalid arguments**.
- For valid commands with correct number of input arguments check the following:
  - The meeting title should be between double quotes "". Otherwise print **Invalid title**
  - Meeting day is an integer from 1 to 365. Invalid day should print **Invalid day**
  - Meeting hour is an integer from 0 to 23. Invalid hour should print **Invalid hour**
  - **The error messages (when applies) should be printed in the following order**:
    Invalid title
    Invalid day
    Invalid hour

For example consider the following input line which has a valid command and correct number of input arguments.

ADD school 100 24

The title is invalid (missing a double quotes) and the hour is invalid (not from 0 to 23), but the day is valid (integer from 1 to 365). Therefore, the planner prints two error messages in the following order

Invalid title

Invalid hour

Note that all messages are case sensitive.

4.2 Input and Output

- The inputs will be provided via an input text file. The file will have commands (ADD,Find,MOD,DEL,Print) each on a separate line.
- The planner should print its outputs (and messages) on the screen.
- Each output/message (whether on screen or in output file) should be on a separate line.
- No empty lines should be left in-between the outputs.
- The names of the input and output text files will be provided as inputs when running the program as follows.

  p2.exe Input.txt

  The above line means that the program should read from Input.txt and print its outputs on the screen.

5 Example Test Cases

Provided with the project example input files In1.txt, In2.txt, and In3.txt

5.1 Example Case 1

Input: p2.exe In1.txt
Output: Conflict 5 10
"Physics Lecture 1" 5 9
"Math Section" 5 10
"Club_meeting" 5 11
"Movies" 100 20
"Math Section"
Empty 7 13
Empty 10 10
"Physics Lecture 1" 5 9
"Math Section" 5 10
"Club_meeting 5 11
"Movies" 100 20
Invalid day
"Physics Lecture 1" 5 9
"Math Section" 5 10
"Club_meeting" 5 11
Invalid hour
Invalid command
"Physics Lecture 1" 5 9
"Math Section" 5 10
"Club_meeting" 5 11
"Lunch" 10 4

The program should print its output on screen.

Figure 2 shows the relation between the input commands and the output lines in case 1 using color boxes

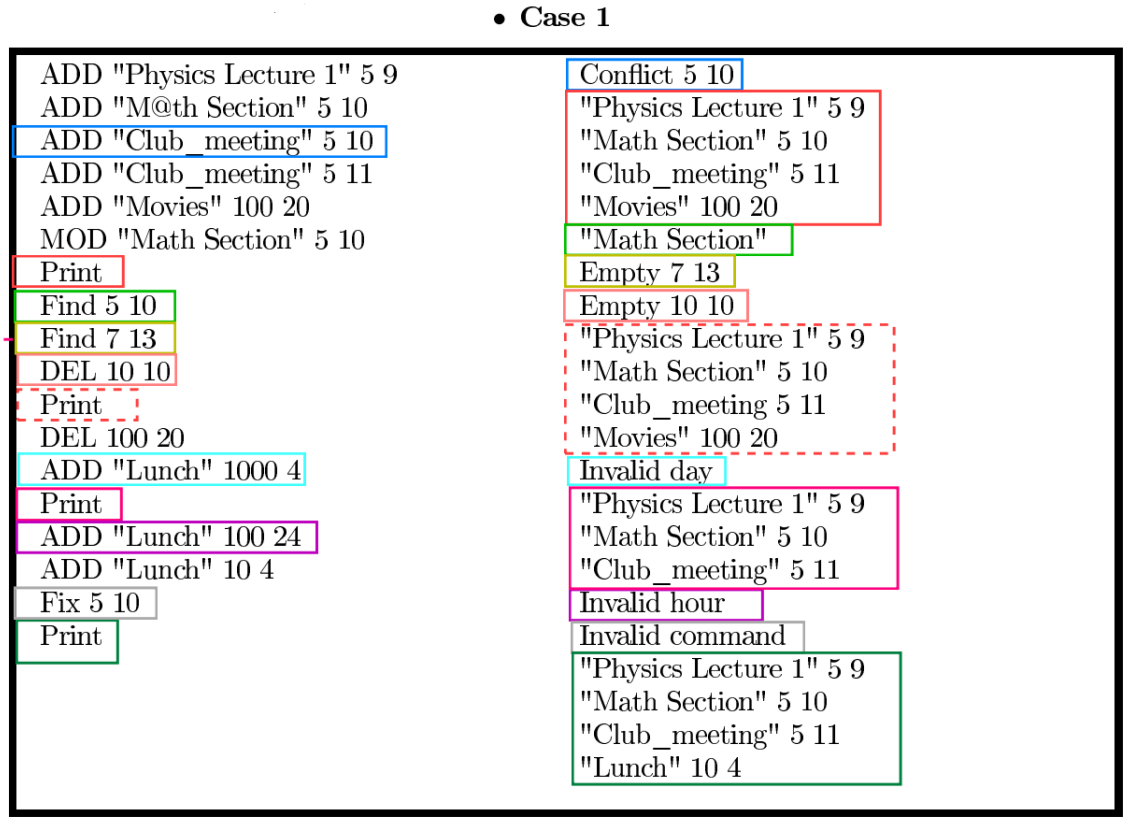

Figure 2: The input/output relation in Case 1

5.2 Example Case 2

Input: p2.exe In2.txt
Output: Empty Planner
Conflict 10 10
"Computer3" 1 0
"Dentist" 3 22
"Physics" 4 2
"Math" 5 10
"Sport" 5 20
"Electronics" 10 10
"Comm1" 360 23
Invalid day
"Computer3" 1 0
"Dentist" 3 22
"Physics" 4 2
"Math" 5 10
"Sport" 5 20
"English class" 10 10
"Communications_1" 360 23
"Physics"
"Computer3" 1 0
"Dentist" 3 22
"Physics" 4 2
"Math" 5 10
"Sport" 5 20
"Library" 8 20
"English class" 10 10
"Communications_1" 360 23

The program should print its output on screen.

Figure 3 shows the relation between the input commands and the output lines in case 2 using color boxes

- **Case 2**

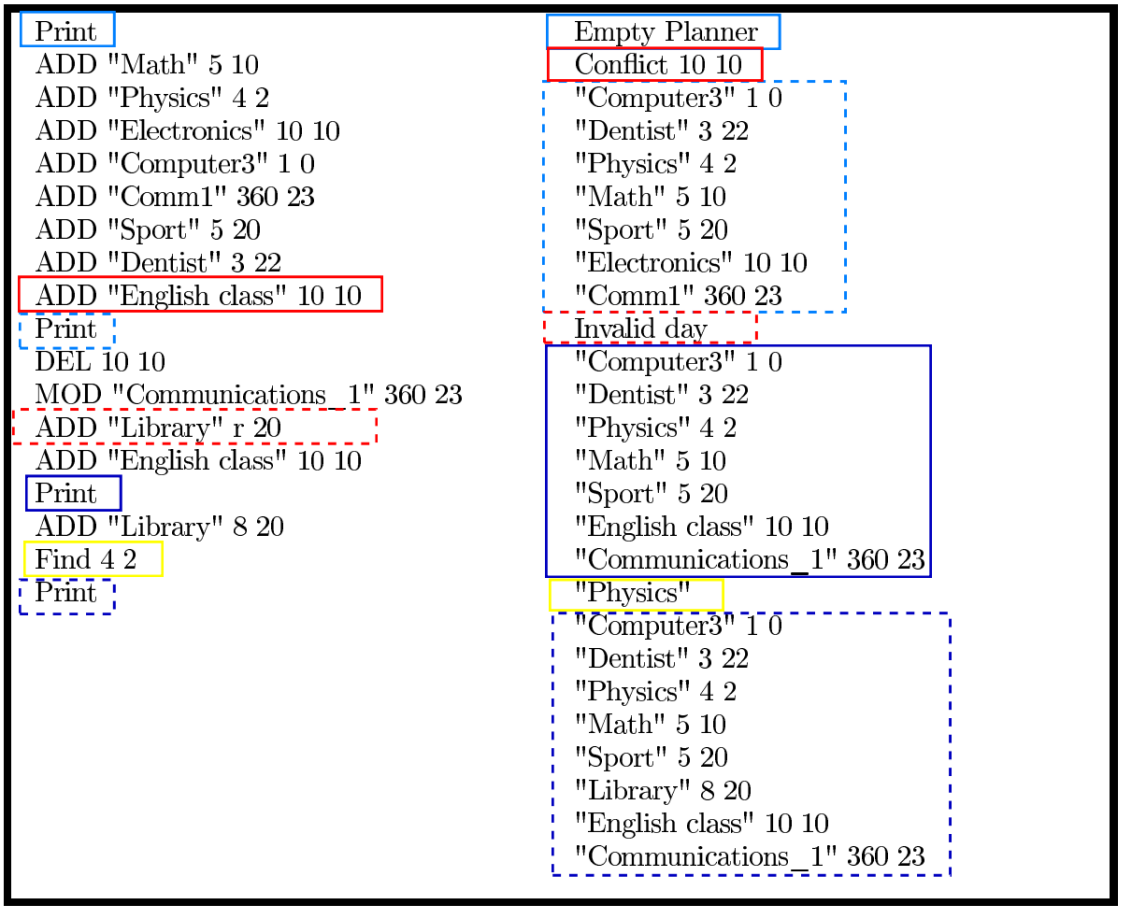

Figure 3: The input/output relation in Case 2

5.3 Example Case 3

Input: p2.exe In3.txt
Output: "Breakfast" 10 7
"Study" 10 8
"Workout" 10 14
"Lunch" 10 16
"Study" 10 17
"watch @ film" 10 20
Invalid command
Invalid hour
"Breakfast" 10 7
"Study" 10 8
"Workout" 10 14
"Lunch" 10 16
"Study" 10 17
"watch @ film" 10 20
"Breakfast" 10 7
"Study" 10 8
"Workout" 10 14
"Lunch" 10 16
"Study" 10 17
"Read a book" 10 20
"Sleep" 10 21
"Breakfast" 10 7
"Study" 10 8
"Workout" 10 14
"Lunch" 10 16
"Study" 10 17
"Sleep" 10 21
Invalid title

The program should print its output on screen.

Figure 4 shows the relation between the input commands and the output lines in case 3 using color boxes

- **Case 3**

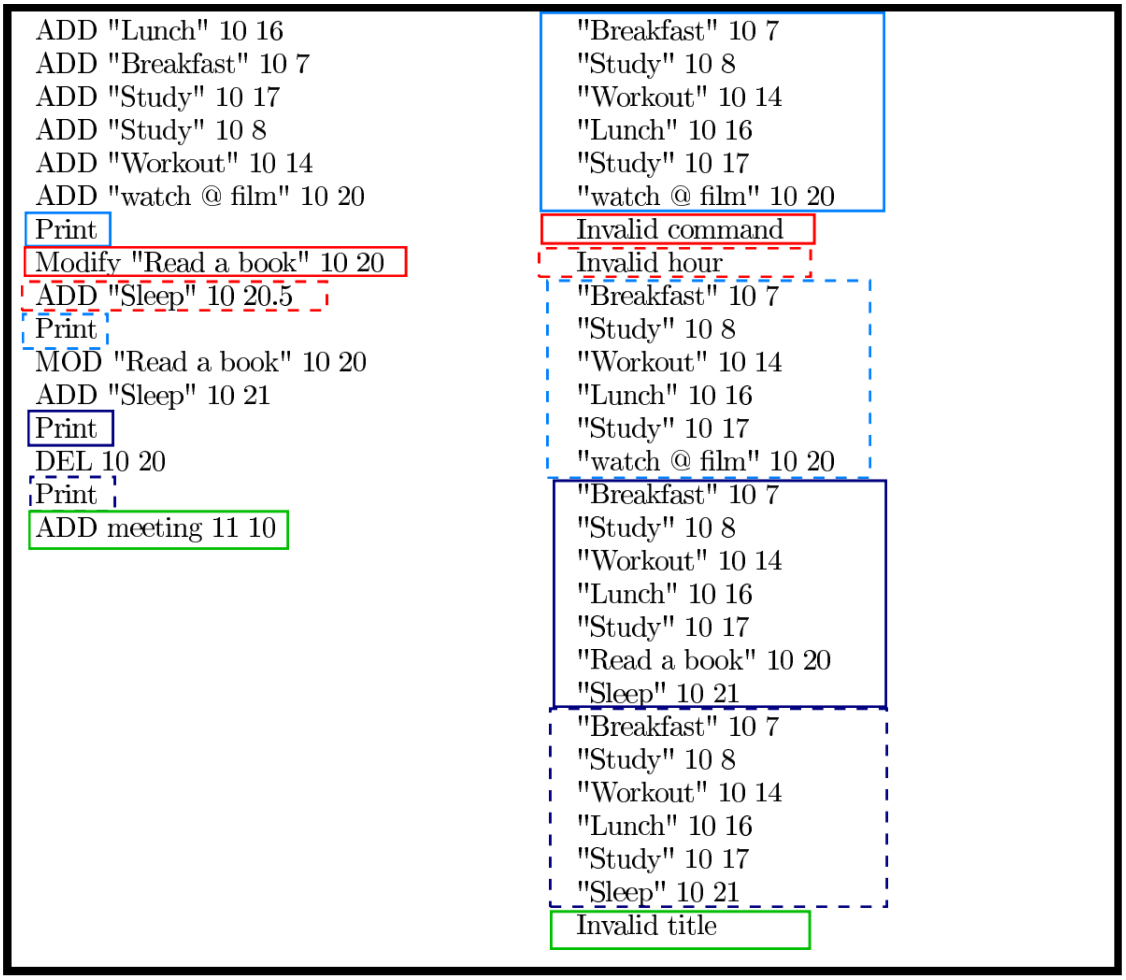

Figure 4: The input/output relation in Case 3