CS 2123 Data Structures

Instructor Dr. Turgay Korkmaz

Homework 2
Due date: check BB
!!!! NO LATE HOMEWORK WILL BE ACCEPTED !!!

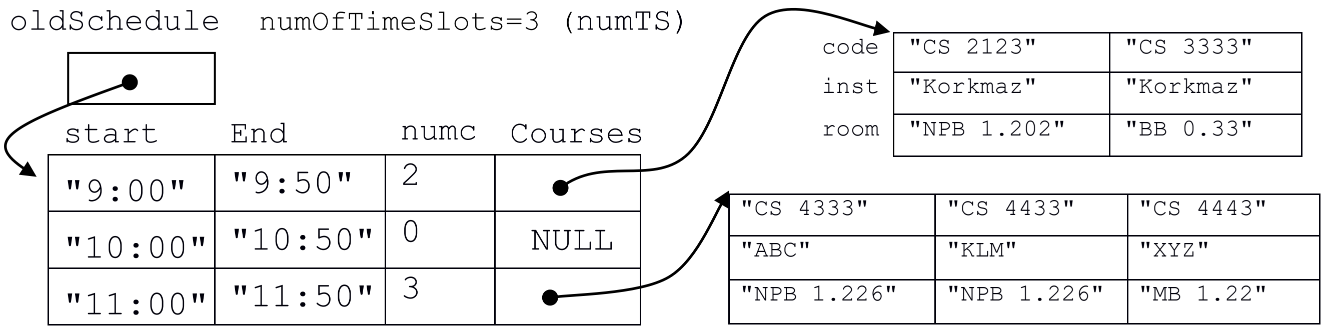
Tips to Save Time:

- Read and understand the assignment before you begin
- Download the provided code and data files under **abc123-hw02** (use your own abc123)
- Make sure you understand the provided code and run it before modifying it
- Review your class notes about File I/O, Dynamic memory allocation/free, Structures, string.h

---

Instructions:

- Familiarize yourself with the provided code which reads an input file and dynamically creates the below data structure and the output when executed as > `hw02 courses-input1.txt`



```
**************Schedule*************
Start   End
9:00    9:50
Course  Instructor      Room
CS 2123 Korkmaz         NPB 1.202
CS 3333 Korkmaz         BB 0.33

Start   End
10:00   10:50
No courses schedule at this time

Start   End
11:00   11:50
Course  Instructor      Room
CS 4333 ABC             NPB 1.226
CS 4433 KLM             NPB 1.226
CS 4443 XYZ             MB 1.22
****************End****************
```

Functions that are already implemented for you, please first understand how they work:

1. scheduleT *LoadSchedule(int numTS);
   This function will load the provided schedule information from an input file given as a command line argument (e.g., "hw02 courses-input1.txt" creates the dynamic structure provided above).
   - **Hints:**
     - fscanf will return an integer the number of successful conversions.
       - checks = fscanf("%d %d %d", &one, &two, &three);
       - if(checks != 3) //then something went wrong
     - fgets reads entire line from file and saves to char[].

2. void PrintSchedule(scheduleT *sch, int numTS);
   This function will print the entire schedule in a format similar to the one provided above.

Functions that you are asked to implement:

3. scheduleT *CopySchedule(scheduleT *oldSch, int numTS);
   This function will take in a pointer for an old schedule and return a pointer to a new copy of that schedule. (C reference card and the string.h will be helpful)

4. void PrintInstructorConflicts(scheduleT *sch, int numT);
   This will iterate through the schedule and print any conflict appearing that has the same instructor assigned to multiple classes at the same time. Print an error message and all conflicting classes. For example, your function should generate the following for the above example:

```
!!!Instructor Conflict!!!
Time: 9:00     9:50
CS 2123   Korkmaz        NPB 1.202
CS 3333   Korkmaz        BB 0.33
```

5. void PrintRoomConflict(scheduleT *sch, int numT);
   This will iterate through the schedule to look for any double bookings of a classroom. If a classroom has 2 or more classes at the same time, print an error message and class information. For example, your function should generate the following for the above example:

```
!!!Room Conflict!!!
Time: 11:00     11:50
CS 4333 ABC     NPB 1.226
CS 4433 KLM     NPB 1.226
```

6. Void FreeSchedule(scheduleT *sch, int numTS);
   This will free all dynamically allocated memory for the schedule pointed by sch. (You will be asked to use valgrind to verify that you freed all the dynamically allocated memory)

Compile/Run:

- To compile, in the command line, type:
  - > gcc -g -Wall hw02.c -o hw02
  - -g: debugging flag creates debugging symbols for use with gdb or ddd debugging tools
  - -Wall: flag alerts all warnings. Helps indicate where code needs to be cleaned or where potential errors can occur
  - -o: place output in file *file*. This applies to whatever sort of output is being produced, whether it be an executable file, an object file, an assembler file or preprocessed C code. If -o is not specified, the default is to put an executable file called a.out. In this case output file is *hw02*
- A Makefile has also been provided. To compile, simply type make in the command line.
- To run program: > ./hw02 course-input1.txt
- To run program and save to file: > ./hw02 course-input1.txt > hw02out.txt
  or copy the terminal output into hw02out.txt file
- Valgrind: > valgrind ./hw02 course-input1.txt
  HEAP SUMMARY will specify bytes in use at exit and number of allocs and frees.
  Proper memory management will say All heap blocks freed - - no leaks are possible
  copy terminal information to hw02valgrind.txt

A minor issue with input files….

When using a Linux machine to work on this program, there may be an issue with the input file that may have microsoft carriage returns that are not read correctly, to check:

1. In terminal, type **`cat input-filename | od -c`**
2. If there are \r characters in the file then the carriage returns are present and must be removed. To remove, run **`sed -i 's/\r//' input-filename`**

What to return: !!!! NO LATE HOMEWORK WILL BE ACCEPTED !!!

1. Do all your work under **abc123-hw02** folder using your own abc123
2. Return your source code and output files in a .zip file (Simply zip abc123-hw02 folder after using “make scratch” to clean up folder of .o and executables)
3. Submit your abc123-hw02.zip via BlackBoard Learn for Class Assignment (Hw-02).

You must submit your work using Blackboard Learn and respect the following rules:

1) All assignments must be submitted as a zip file unless it is a single pdf file.
2) Assignments must include all source code.
3) Assignments must include output files demonstrating the final test output run by the student.
4) If your assignment does not run/compile, the output.txt file should include an explanation of what was accomplished, what the error message was that prevented the student from finishing the assignment and what the student BELIEVES to be the underlying cause of the error.
