Assignment 1 - part 1

Deadline	**Anytime before Sunday, 18th March 2018, 15:00**
Evaluation	10 marks –which is 5% of your final grade
Late Submission	5% per hour (or fraction of hour) it is late
Team	The assignment can be done individually or in pairs (of at most 2 students)
Purpose	Practice with C++ input and output, basic constructs and structures.

Problem to solve

INMS has a number of rooms available for teaching and meeting. Staff can book a room for the appropriate purpose, usually for no more than the time allowed for each room. The booking requests are not confirmed immediately but must be approved by INMS administration. Upon reviewing the booking requests and room availability, the admin can decide to approve, decline, or suggest an alternative. Knowing that, some staff make requests without specifying the rooms they need, but instead leave the decision to the admin.

In this assignment, you are writing a program that helps INMS expedite the approval process. Room descriptions are stored in a file name `rooms.txt`, and requests for bookings are in `bookings.txt`. Your program will read these two files and output any inconsistency it encounters.

Name your program `a1p1.cpp`
Hand-in: Submit `a1p1.cpp` electronically using the submission form on STREAM.

If you have any questions about this assignment, please ask the lecturer before the assignment is due.

Problem statement

Write a program that does the following:
1. Displays on the standard output information about the authors of the program (IDs , family & given names, assignment number)
2. Reads data contained in the files `rooms.txt` and `bookings.txt`, as necessary.
3. If there is any problem reading the files, stops immediately and outputs an error message to the standard error stream.
4. If both files are read successfully, writes out to the standard output any inconsistencies it discovers

Your program should perform the following consistency checks:
1. Is there any room that is not booked?
2. Is there any booking that does not specify a room?
3. Is there any booking of the wrong type, for example: a meeting room booked for teaching,

or vice versa?
4. Is there any booking that is longer than the maximum allowed time, for example: book 180 minutes for a room that allows maximum 60 minutes?

For each category of inconsistency, output the list of rooms/bookings failing a consistency check.

Input and output

The first line in `rooms.txt` specifies the total number of available rooms.
Similarly, the first line in `bookings.txt` specifies the total number of the booking requests made.
After that, every 3 lines in `rooms.txt` describe information about one room. The first line is the name of the room, the second line is the purpose of the room, and the last line is the maximum time allowed in minutes.
Similarly, every 4 lines after the first in `bookings.txt` describe information about the booking request. They are in order: name of the staff who made the request, purpose of the booking, name of the room (or `---` if unspecified), length of booking in minutes.

You can assume that:

- The files are written in pure ASCII, no Unicode or strange characters.
- There is no trailing space (including whitespaces, tabs, and newline characters) anywhere in the file. Each field is on its own line, that's it.
- There are only two possible values for `purpose:` `teaching` and `meeting`
- Each file contains at most 100 rooms or bookings

Sample input

> *BE AWARE: the following input files resemble but are not the ones used to mark your assignment. This will be the procedure for all 159.234 assignments.*

File rooms.txt

```
4
INMS-Lab1
teaching
60
INMS-Lab2
teaching
120
INMS-2.14
meeting
60
INMS-3.08
meeting
120
```

File bookings.txt

```
8
cscorgings
teaching
INMS-Lab1
60
ecalude
teaching
INMS-Lab1
60
dpplayne
teaching
INMS-Lab2
120
agilman
meeting
INMS-Lab2
60
```

```
ecalude
meeting
INMS-3.08
180
jren
teaching
INMS-3.08
120
tliu
teaching
---
60
mpawley
teaching
---
60
```

Sample output

Given the input files above, your program should produce the following output-make sure the **formatting** of your output is **well structured** and follows **EXACTLY** the one here:

```
************************************************
* 159.234 Assignment 1 Part 1
* Name: Jane Doe, John Smith
* ID: 17012345, 16234567
************************************************
The following rooms have not been booked by anyone:
Room: INMS-2.14 suitable for meeting purpose allows maximum 60 minutes

The following booking requests do not specify a room:
Booking: by tliu for teaching purpose at room --- for 60 minutes
Booking: by mpawley for teaching purpose at room --- for 60 minutes

The following booking requests have the wrong purposes:
Booking: by agilman for meeting purpose at room INMS-Lab2 for 60 minutes
Booking: by jren for teaching purpose at room INMS-3.08 for 120 minutes

The following booking requests are over the time limit:
Booking: by ecalude for meeting purpose at room INMS-3.08 for 180 minutes
```

If for example, file `rooms.txt` is missing, your program must print out EXACTLY[1] the following:

```
************************************************
* 159.234 Assignment 1 Part 1
* Name: Jane Doe, John Smith
* ID: 17012345, 16234567
************************************************
Error: file rooms.txt is missing. Program terminates.
```

Your program will be compiled, run and evaluated by a machine. So besides your name(s) and ID(s), the output must be EXACTLY the same, character to character.

[1] Of course your names and IDs should be used instead of Jane Doe…..

Requirements

What you must and must not do

1. Your program MUST be named `a1p1.cpp`. If you use a different name, you will get zero.

2. Your solution MUST contain at least two struct types, one called `Room` that represents a single room, and another called `Booking` that represents a single booking.

3. You MUST read the information from the input files into an array of Bookings and an array of Rooms. These arrays must not be global.

4. Each file MUST only be read ONCE.

   This means you are not allowed to repeatedly read from the input files over and over in order to perform the consistency checks.

5. Your program MUST NOT require any user interaction.

   This means you cannot ask the user to press any key, or type in anything. Your program should read the input files and perform the consistency check automatically. When your program finishes printing out the inconsistency, TERMINATE IMMEDIATELY.

6. Your program SHOULD be organised as followed:
   - Header (Your name(s), ID (s), short description for the program, etc)
   - All included files/libraries
   - Struct declarations
   - Function prototypes
   - Main function
   - Function definition

7. All functions and structs MUST be properly documented following Doxygen format. See http://www.yolinux.com/TUTORIALS/LinuxTutorialC++CodingStyle.html for detail. You will also learn how to document your code in the tutorials.

8. You MUST NOT use classes

9. You MUST NOT use dynamic memory allocation, or raw pointers.

   This means you cannot use the `*` *operator, but reference is OK. i.e.* `int *i` *is forbidden, but* `int &i` *is fine.*
   All arrays must be instantiated as fixed length.
   Any keyword that invokes memory allocation will earn you a zero. They include but not limited to: `new`, `delete`, `malloc`, `free`, …

10. You MUST NOT use advanced features not covered by the course by the due date, e.g. linked lists, maps, vectors, sets, deques, … You will be penalised in case you do use them.

Miscellaneous

1. When working in pair, send one solution file per team.

2. The assignment will be discussed on Wednesday lecture before the assignment is due and solutions will be discussed on Monday lecture after the due time.

3. Marks will be allocated for: correctness, completeness, use of C++ constructs presented in class/tuts, simple and clear solution, good documentation, and structured output display (on screen).

4. Using goto, non-constant global variables or C-like I/O constructs (i.e printf fprintf, scanf, FILE*,etc) is not allowed and it will be penalised. Only const global variables are allowed.

5. Programs that do not run or do not compile in the (Albany) labs, using gcc(SciTe), get 0 marks.

6. The program must be your own work. Please be aware that you might be asked to explain to your lecturer how your program works. If you cannot explain it, then it is not yours and you will get 0 marks for that assignment. Attributing someone else's work as your own is plagiarism, and it is a violation of Massey University policy. We might file an official complaint against any student who we believe has committed plagiarism.

7. Suspicious similar solutions will all get 0 marks-see also point 7 above