Highlighted = completed

CS174 – F2016 – Final Project

**BACKGROUND**

This project will simulate the daily activity of a hotel. This activity includes guests checking in, checking out, and purchasing items and services. Any purchases will be put on the room’s bill. Purchases may include, but are not limited to drinks, food, items from the hotel shop, an extra cot, spa services.

This project will be written in C++ and must run on CodeBlocks on a school issued laptop. See rubric for grading details.

**OPERATION**

*Hotel*

A hotel has a number of floors. Each floor has the same number of rooms. Your program will read in qualities of the hotel from a configuration file called “hotel.conf”. This file will contain the name, address, phone numbers, amenity list, and other information from which to construct a hotel object.

*Actitvity*

Once the hotel is contructed, your program must begin to read commands from a file. These commands are: check in, check out, purchase, bill, report. The activity file will be called “hotel.act”

**Check In**: If there is a vacancy assign the guest a room. If there is no vacancy, ignore the request. All names will be unique and guests may only reserve one room per check in.

FORMAT: CHECK\_IN: <Name>, <Number in Party>, <Date>

**Check Out**: Create a file using guest’s name and check out date as the file name and “.bill” as the file extension. For example, if the Murray guest checks out on January 12, 2017, the name of the file would be Murray12012017.bill If there is already a file with that name, you may overwrite. This file will contain the hotel’s information, guest’s name, room ID, check in date, check out date, and an itemized list of charges (each day is one charge) as well as tax charge. This operation will also free up the guests’s room. If the guest is not checked in, ignore the command. If check out date is before check in date, ignore the command.

RATE NOTE: There are two different rates: Peak and Off Peak. Be sure to give correct charges, especially if a guest’s stay spans the two different rates.

DATE NOTE: Dates are in format DDMMYYYY

FORMAT: CHECK\_OUT: <Name>, <Date>

**Purchase**: A purchase is a charge added to the bill. Each purchase will have a name, description, and fee. If the name on the charge is an actual guest at the hotel, add the charge to the guest’s bill. If an un-registered guest makes a charge, the charge should be recorded in a list of non-guest purchases.

FORMAT: CHARGE: <Name>,<Description>, <Amount>,<Date>

**Bill**: Creates a file using the guest’s name and given date as the file name and “.bill” as the file extension. For example, if the Murray party generates a bill on January 10, 2017, the name of the file would be Murray10012017.bill . The guest’s room is not freed up… This is just a bill generation.

FORMAT: BILL:<Name>, <Date>

**Report**: Generates a file with the name of the hotel and the date along with the list of current guests, number in their party, and the list of all non-guest purchases of the current day.

FORMAT: REPORT:<Date>

**ROBUSTNESS**

You may assume that the commands in the activity are *syntactically* correct. For instance, you can assume that all CHECK\_IN commands will indeed include all needed parameters in the correct order.

Your code must make semantic checks: ie., a check out command for a guest who is not checked in, a charge for a date for which the guest is not checked in, etc..

**FILES**

Included with this project description, you’ll get a sample hotel.config, sample hotel.act, as well as a sample .bill.

**DRIVER**

Your driver will have no user input. It will read the hotel.conf file, then will operate and create files as directed by the hotel.act file.

**ASSIGNMENT**

Using what you know about good object design, the C++ language, and the STL (Standard Template Library) write the code for this project. See the rubric for grading weights. We will be going over some object designs in class for you to use as a basis.

I need this to split strings:

string commandFileName = "hotel.act";

string line;

ifstream commandFile(commandFileName.c\_str());

if (commandFile.is\_open())

{

while (getline(commandFile,line))

{

cout<<"Line: "<<line<<endl;

istringstream ss(line);

string token;

while(getline(ss, token, ',')) {

cout << "token: "<<token << '\n';

}

cout<<"----------------------"<<endl;

}

commandFile.close();

}