In this machine project, you will write a program for a simplified **room reservation system** for a hotel establishment. The aim of this project is for you to make use of various concepts learned in **CCPROG2**, including arrays, structures/records and file input/output.

A. Overview

The "room reservation" program will: (a) accept a series of booking requests from **`stdin`**; (b) check and book these requests against room availability - room information comes from an input text file "**`rooms.txt`**"; (c) generate the outcome of these bookings via **`stdout`**; (d) generate booking confirmations for successful bookings - one output text file for each successful booking request; and (e) just before exiting, the program will generate an occupancy report "**`report.txt`**" for the management of the hotel.

Before accepting any booking requests, the program has to be read information about the rooms from an input file "**`rooms.txt`**". This file contains the # of rooms, type of the rooms, and relevant pricing. This input file must have a minimum of one line of data; each line representing one type of room and its relevant information. You will then update internal variables, arrays or structures or combinations of such, with the information from this input file. This will be used to track room availabilities as the program enters into the booking request processing phase.

B. Room Input Data

The input "**`rooms.txt`**" text file contains the information about the various available rooms for booking. Each line within this file represents a type of room, along with the # of available rooms for this type and the rate (in PHP) for booking that room. For simplicity, we will assume that a room is booked on a per-day basis, rather than the usual per night. Specifically, each line in this input file has these fields and in this exact order:

field name	format	meaning
room code	`char`	the single letter for the room (must be **unique** for all lines and valid only for letters '**A**' to '**Z**')
room capacity	`int` (from 1 to 12)	how many guests can fit the room
# of rooms	`int` (from 1 to 40)	the number of rooms in the hotel with this capacity
weekday rate	`float` (d+.dd)	the rate per day during Mondays to Friday
weekend rate	`float` (d+.dd)	the rate per day during weekends (Saturday and Sunday)
per guest rate	`float` (d+.dd)	fee to add per day per guest beyond the first guest (applicable only if the room has a capacity of 2 or more)

Study the example **`rooms.txt`** file below to further understand how the information is used:

Sample "rooms.txt" file	Interpretation of the values

`A  1 3 1500.00 2500.00   0.00`
`B  1 2  800.00 1000.00   0.00`
`C  2 2 2000.00 3000.00 300.00`
`D  4 2 4000.00 5000.00 400.00`
`K 10 1 7000.00 7000.00 500.00`
	three 'A' rooms, each for single occupancy
	two 'B' rooms, each for single occupancy
	two 'C' rooms for up to two guests per room
	two 'D' rooms can have up to four guests each
	one 'K' room with capacity for up to ten guests

The above sample input file has **five** lines; each line describes a different type of room with its unique room code; along with the information about the room cost per day. This input file describes the hotel as having ten (**`10`**) rooms (3 + 2 + 2 + 2 + 1). Notice the use of whitespaces as field separators; and you can also assume that the first character of each line is always the single letter room code, i.e., there are no leading white spaces per line.

The fourth and fifth fields are used for calculating the cost per day (depending on whether it is a weekday or weekend) for the room and per guest rate is applied for each guest above the first one. For example, if four guests were to book a one-day stay on a weekday for room D, the cost will be 4,000.00 + 3 × 400.00 = 5,200.00. Room '**`D`**' costs 4,000 per weekday and because we have four guests, three of the guests will each will need to pay 400 pesos per day, hence the calculation of 3 × 400.00 = 1,200.00.

Carefully study the following additional examples:

If a booking request is for one room with:	**Total cost**
one guest staying on two days in an '**`A`**' room: Sunday & Monday two guests staying for three days in a '**`C`**' room: Monday to Wednesday four guests staying for two days in a '**`D`**' room: Saturday and Sunday	1,500 + 2,500 = 4,000 3 × (2,000 + 1 × 300) 2 × (5,000 + 3 × 400)

A few notes about this input file is in order:

1. If the capacity of a room is 1, the per guest cost is **`0.00`**.
2. The order of the rows does not matter. For example, this input file can also be written as:

   ```
   C  2 2 2000.00 3000.00 300.00
   K 10 1 7000.00 7000.00 500.00
   B  1 2  800.00 1000.00   0.00
   A  1 3 1500.00 2500.00   0.00
   D  4 2 4000.00 5000.00 400.00
   ```

   and the program should behave in the same way as with the original version.
3. The room rates (fourth and fifth fields) cannot be `0.00`.
4. The price fields (4th, 5th & 6th fields) have **two** decimal digits after the decimal point.

C. Booking Requests

A booking request is for exactly one room and has the following data fields:

field name	format (maxlen)	meaning
last name	`string` (25)	last name of the main guest (one word only)
first name	`string` (25)	first name of the main guest (one word only)
email address	`string` (40)	email address of the main guest
room code	`char`	the code for the room type being booked
total # of guests	`int` (dd)	how many guests for this booking, this is checked against the size of room selected by the room code
starting **DoW**	`int` (d)	the **D**ay **o**f **W**eek for the **`first`** day of the stay: the convention is `0` - `monday`, `1` - `tuesday`, `2` - `wednesday`, …, `5` - `saturday`, `6` - `sunday`
# of days to stay	`int` (d)	**minimum** of `1` **maximum** of `7`

Note that if the starting DoW is Sunday, then the booking can only be for 1 day, which is Sunday itself. This is because this booking system offers only **one** week of availability, starting from Monday. So if the booking requests to start the stay on a Monday, the # of days can be from 1 to 7. Seven days of stay means staying for the entire week, Monday to Sunday.

D. Sample Program Run

Now we can show a sample run of the program. Please study the behavior of the program. The user input is printed in **`boldface`**. The `<ENTER>` represents pressing the enter key. Please note that all names and email addresses are fictional.

```
Welcome to the hotel booking system.

Loading hotel information from rooms.txt… Success!
The hotel has 10 rooms available.

Enter booking request #1:
  last name: Smith
  first name: John
  email address: johnsmith@example.email.com
  room code: D
  total # of guests (max 4): 4
  starting day: 5
  # of days to stay (max 2): 2

Successfully booked a room for 4 guest/s for 2 day/s of stay starting Saturday
Total amount for this booking in PHP: 12400.00

Enter booking request #2:
  last name: Taylor
  first name: Sweeeft
  email address: staylor@example.email.com
```

```
  room code: K
  total # of guests (max 10): 8
  starting day: 3
  # of days to stay (max 4): 3

Successfully booked a room for 8 guest/s for 3 day/s of stay starting Thursday
Total amount for this booking in PHP: 31500.00

Enter booking request #3:
  last name: Prince
  first name: Diana
  email address: wonderwoman@example.email.com
  room code: A
  total # of guests (max 1): 1
  starting day: 0
  # of days to stay (max 7): 7

Successfully booked a room for 1 guest/s for 7 day/s of stay starting Monday
Total amount for this booking in PHP: 12500.00

Enter booking request #4:
  last name: Luthor
  first name: Lex
  email address: supervillain@example.email.com
  room code: K
  total # of guests (max 10): 6
  starting day: 2
  # of days to stay (max 5): 2

Sorry, I am unable to satisfy this request due to room availability

Enter booking request #5:
  last name: Luthor
  first name: Lex
  email address: supervillain@example.email.com
  room code: C
  total # of guests (max 2): 2
  starting day: 2
  # of days to stay (max 5): 3

Successfully booked a room for 2 guest/s for 3 day/s of stay starting Wednesday
Total amount for this booking in PHP: 6900.00

Enter booking request #6:
  last name: Bond
  first name: James
  email address: agent007@example.email.com
  room code: C
  total # of guests (max 2): 2
  starting day: 4
  # of days to stay (max 3): 3

Successfully booked a room for 2 guest/s for 3 day/s of stay starting Friday
Total amount for this booking in PHP: 8900.00

Enter booking request #7:
  last name: Kirk
  first name: James
  email address: captain@example.email.com
  room code: C
```

```
  total # of guests (max 2): 2
  starting day: 3
  # of days to stay (max 4): 3

Sorry, I am unable to satisfy this request due to room availability

Enter booking request #8:
  last name: Mouse
  first name: Mickey
  email address: MM90birthday@example.email.com
  room code: D
  total # of guests (max 4): 3
  starting day: 3
  # of days to stay (max 4): 3

Successfully booked a room for 3 guest/s for 3 day/s of stay starting Thursday
Total amount for this booking in PHP: 15400.00

Enter booking request #9:
  last name: Duck
  first name: Donald
  email address: dduck@example.email.com
  room code: B
  total # of guests (max 1): 2
  starting day: 1
  # of days to stay (max 6): 3

Sorry, I am unable to satisfy this request due to room capacity

Enter booking request #10:
  last name: Danvers
  first name: <ENTER>
  email address: supergirl@superheroes.email.com
  room code: F
  total # of guests (max 0): 1
  starting day: 1
  # of days to stay (max 6): 3

Sorry, I am unable to satisfy this request due to invalid room code

Enter booking request #11:
  last name: Danvers
  first name: <ENTER>
  email address: <ENTER>
  room code: A
  total # of guests (max 1): 1
  starting day: 5
  # of days to stay (max 2): 2

Successfully booked a room for 1 guest/s for 2 day/s of stay starting Saturday
Total amount for this booking in PHP: 5000.00

Enter booking request #12:
  last name: ThePooh
  first name: Winnie
  email address: <ENTER>
  room code: D
  total # of guests (max 4): 1
  starting day: 7
  # of days to stay (max 0): 3
```

```
Sorry, I am unable to satisfy this request due to invalid starting day.

Enter booking request #13:
  last name: ThePooh
  first name: Winnie
  email address: pooh@example.email.org
  room code: D
  total # of guests (max 4): 1
  starting day: 5
  # of days to stay (max 2): 3

Sorry, I am unable to satisfy this request due to invalid # of days to stay

Enter booking request #14:
  last name: <ENTER>

Processed 7 successful requests out of 13 submitted requests.
Generating management report.
Thank you for using the hotel reservation system.
Bye!
```

E. Input Validation for the "rooms.txt" file

Please note these observations from the above sample run:

1. At startup, program initializes some internal variables by reading the **`rooms.txt`** file. If this file is missing or unreadable, the program will output the following and exit:

   ```
   Loading hotel information from rooms.txt… Failed!
   The rooms.txt file is unreadable or missing.
   Exiting the program. Bye!
   ```

2. In the above run, the **`rooms.txt`** lists a total of `10` available rooms on **five** lines. If for some reason, the # of available rooms adds to `0` (zero) after reading "**`rooms.txt`**", the program will also exit with this output. This is also the error message if the input file has `0` lines.

   ```
   Loading hotel information from rooms.txt… Success!
   The hotel has 0 rooms available.
   No eligible rooms were found in the input file, exiting the program. Bye!
   ```

3. The room code has to be **unique** for the entire file. If the input file has two or more lines with the **same** room code, the program will also exit with this output. This means you need to check if the "room code" is **unique** within the file, and produce this error if this is not the case.

   ```
   Loading hotel information from rooms.txt… Failed!
   The hotel "room codes" are not unique!
   Exiting the program. Bye!
   ```

4. The room code cannot be any other letters but '**A**' through to '**Z**'. Lower-case letters are **invalid**. Decimal digits '0'-'9' and punctuation symbols are also **invalid**. Generate the error message below if the program encounters an invalid room code. This means that

there can only be 26 different room types. Display also the line number with # of the line in the file that has the invalid room code.

```
Loading hotel information from rooms.txt… Failed!
Found an invalid "room code" at line #!
Exiting the program. Bye!
```

F. Further Information on Booking Requests

For booking requests, please observe these:

1. After successfully reading the **`rooms.txt`** file, the program **always starts** at booking `#1`. At this point, all rooms are available for all seven days.
2. The program enters into a loop to accept booking requests, incrementing the booking # by one each time **even** if the booking was unsuccessful. It also keeps a count the total number of requests and also the number of successful requests. These two values are printed at the end before the program prints out the last two lines: "**`Thank you`**…" and "**`Bye!`**".

   ```
   Processed 7 successful requests out of 13 submitted requests.
   ```
3. A booking request is always for **one** room; and a room can be booked for a **minimum** of **one** day and **up to seven** consecutive days (exactly one week), depending on which **DoW** was the first day of stay. For this project, the rooms are only available for seven days (starting on Monday and ending on Sunday).
4. Assume that the last name, first name and email address are single words, that is, they do not contain any spaces, tabs, etc. However, you are free to explore more than a word-length for your last name, first name and email addresses. Do note that rules for the correct email address format should be followed.
5. The **last name** field **cannot** be empty -- this is **important** as you will see later as you read through the rest of this document. This means that the program will accept a booking even when the **first** name is an empty string. This is also the case with the **email** address, which can also be empty. See booking `#10` and `#11` as examples. In the case of `#10`, the booking was unsuccessful as the room code '**`F`**' is invalid, i.e., it is absent on the **`rooms.txt`** file.
6. If a valid room code is entered, the next line will display the maximum capacity of the room. For example, in booking request `#6`, room code **`C`** refers to the third line in the **`rooms.txt`** file, which has a maximum capacity of 2 guests per room. Therefore the program outputs "**`(max 2)`**" below:

   ```
   total # of guests (max 2):
   ```

   Likewise, study the other booking examples above to see the behavior of the program. Notice that if the room code is invalid, the max value is **zero** (**`0`**).
7. Study bookings `#12` and `#13` for the error messages if the starting day of week (**`DoW`**) is invalid (`#12`) or if the # of days to stay is invalid (`#13`). The valid starting **`DoW`** is a value

from **0** to **6**, with **0** to mean Monday and **6** to mean Sunday. Other invalid input or bookings should be considered as well.

8. If a valid **DoW** is entered, the next line will display:

```
total # to stay (max X):
```

where **X** is a value calculated depending on the **DoW** entered. Study the example bookings above and see the pattern. Hint. **X** is a value between **0** and **7**, inclusive.

9. The end of all input is indicated by entering an empty **last name**. This is simply done by an typing **<Enter>** or **<Return>** key when the last name is being requested by the program, as shown in booking **#14**.

G. Program File Output For Each Successful Booking Request

1. For **each** successful booking request, the program will generate an output "booking confirmation email" text file. The program will **NOT** be able send out any emails, and instead it will generate output files to "mimic" or simulate sending a confirmation email to the the customer.
   The format of this text file is:

```
Dear <last name in ALL CAPS ①> <first name ②>,

Thank you for booking your holiday with us.
Your booking reference number is: <booking request number ❶>
The email address for this booking: <email address ❷>

Booking details:
     Room Code: <room code ❸>
     Number of guests: <# of guests ❹>
     First day of booking: <dayofweek in words ❺>
     Number of days: <# days of stay ❻>
     Number of weekday stay: <# days that fall within Mon-Fri ❼>
     Number of weekend stay: <# of days that fall within Sat and/or Sun ❽>

Fees:
     Weekday room rate: <weekday rate per day in PHP ⑩>
     Weekend room rate: <weekend rate per day in PHP ⑪>
     Total room cost for weekday stay: <total cost for weekday stay ⑫>
     Total room cost for weekend stay: <total cost for weekend stay ⑬>
     Number of guests above the first: <# of guests - 1 ⑭>
     Total per guest cost: <per guest cost for the entire stay ⑮>
     Total guest cost for the stay: <per guest cost × (#guests - 1) ⑯>

     Total: <total costs for the entire stay ⑰>

Have a nice day! Hope you have a pleasant stay.
```

On the template / format above, the text printed in **<boldface ❾>** are the fields that will be generated for each text file. A tag, e.g., a circled number like ❾ or ⑰**,** is provided on each output field on the template above for easy reference in the following discussion:

- ① -- this is the last name listed on the booking request, displayed in uppercase
- ② -- this is the first name listed on the booking request, if available
- ❶ -- this is the booking request number, e.g., **1**, **2**, etc.
- ❷ -- this is the email address provided by in the booking request, if available
- ❸ -- this is the room code like **A**, **B**, etc.
- ❹ -- this is the number of guests for the booking, this number is at least **1**
- ❺ -- display `Monday` if the booking was to start on Monday (day 0), etc.
- ❻ -- the number of days for this booking, note that **0** < ❻ <= **7** and ❻ == ❼ **+** ❽
- ❼ -- the number of **weekdays** out of ❻, note that **0** <= ❼ <= **5**
- ❽ -- the number of **weekend** days out of ❻, note that **0** <= ❽ <= **2**
- ⑩ -- the single day weekday room rate for the room of code ❸
- ⑪ -- the single day weekend room rate for the room of code ❸
- ⑫ -- this is computed as: ⑫ = ❼ × ⑩
- ⑬ -- this is computed as: ⑬ = ❽ × ⑪
- ⑭ -- this is simply ⑭ = ❹ - **1** and this number can be **0**
- ⑮ -- computed as: ⑮ = ❻ × ⑭
- ⑯ -- computed as: ⑯ = ⑭ × ⑮

2. The program will generate one text file for **each successful** booking. The filename format to use for each text file is:

   ***<last name in all caps>-<booking#>*.txt**

   Please ensure that you strictly follow this file naming convention. For booking request #6 by "**James Bond**", the generated text file will have this filename: "**BOND-6.txt**". Notice that during a run of the program, it is *impossible* to have two or more bookings writing into the same output file, since the **booking#** is unique on each run.
   Obviously, if a booking was **unsuccessful**, like booking **#12**, **NO** output file is generated. On the example run above, there were **seven** (7) successful bookings, so there will be **seven** output files: "**SMITH-1.txt**", "**TAYLOR-2.txt**", until "**DANVERS-11.txt**".

3. Below would be the text file output for booking **#6** on the given sample run above. The filename for this output file will be: "**BOND-6.txt**".

The fields are printed in **boldface** for emphasis; obviously on the ASCII text file output, it is not necessary to print these in boldface.

```
Dear BOND James,

Thank you for booking your holiday with us.
Your booking reference number is: 6
The email address for this booking: agent007@example.email.com

Booking details:
     Room Code: C
     Number of guests: 2
     First day of booking: Friday
     Number of days: 3
     Number of weekday stay: 1
     Number of weekend stay: 2

Fees:
     Weekday room rate: 2000.00
     Weekend room rate: 3000.00
     Total room cost for weekday stay: 2000.00
     Total room cost for weekend stay: 6000.00
     Number of guests above the first: 1
     Total per guest cost: 900.00
     Total guest cost for the stay: 900.00

     Total: 8900.00

Have a nice day! Hope you have a pleasant stay.
```

4. Assume that each time the program is run, all previously generated output files have been deleted.
5. It is recommended that you write your program logic so that each output file is generated as soon as the booking request is successfully processed. For example, around the point when it produces this **`stdout`** output for booking **#8**:

```
Successfully booked a room for 3 guest/s for 3 day/s of stay starting Thursday
Total amount for this booking in PHP: 15400.00
```

your program should also generate the corresponding output file: "**`MOUSE-8.txt`**". This applies for all other successful bookings, simply process the booking request and generate the output file before going to accept the next booking request.

H. Program File Output after Processing all Bookings

Once the user provides an empty string (by pressing the <enter> or <return> key) during the prompt for the last name, the program will now enter its final / completing phase of execution. At this point, it displays:

```
Processed X successful requests out of Y submitted requests.
Generating management report.
Thank you for using the hotel reservation system.
Bye!
```

The following must be kept in mind:

1. Your program will need to keep track of the number of successfully completed booking requests; and also the total # of booking requests. The first value is to be printed where the X is. The second value is where the Y is.

   ```
   Processed X successful requests out of Y submitted requests.
   ```

   Notice that in the sample run above, **X** is 7 and **Y** is 13.
2. The program now enters into the management report generation phase. The filename is for this output ASCII text file is: "**`report.txt`**". The format of this report is as follows:

   ```
   Room Reservation Report

   1, <rcode_1>, <wdrtcnt_1>, <wdrtbkd_1>, <ssrtcnt_N>, <ssrtbkd_N>, <%occ_1>
   2, <rcode_2>, <wdrtcnt_2>, <wdrtbkd_2>, <ssrtcnt_N>, <ssrtbkd_N>, <%occ_2>
   3, <rcode_3>, <wdrtcnt_3>, <wdrtbkd_3>, <ssrtcnt_N>, <ssrtbkd_N>, <%occ_3>

   <N>, <rcode_N>, <wdrtcnt_N>, <wdrtbkd_N>, <ssrtcnt_N>, <ssrtbkd_N>, <%occ_N>
   ```

   The **<N>** above represents the number of room codes, and this is also the same as the number of lines on the "rooms.txt" input file. The other fields are as follows:
   - <rcode_k> -- the room code for the kth line on the rooms.dat file
   - <wdrtcnt_k> -- the total # of rooms on the five weekdays for this room code
   - <wdrtbkd_k> -- the total # of rooms that were booked for these five weekdays
   - <ssrtcnt_k> -- the total # of rooms on the Saturday and Sunday for this room code
   - <ssrtbkd_k> -- the total # of rooms that were booked for both Saturday and Sunday for this room code
   - <%occ_k> -- % of occupancy, and this is computed as:

     (<wdrtbkd_k> + <ssrtbkd_k>) / (<wdrtcnt_k> + <ssrtcnt_k>)

   The order of the room codes should follow that on the "rooms.txt" file.
3. As an example, this will be the report for the above sample run:

   ```
   Room Reservation Report

   1, A, 15, 5, 6, 4, 42.86%
   2, B, 10, 0, 4, 0, 0.00%
   3, C, 10, 4, 4, 2, 42.86%
   4, D, 10, 2, 4, 3, 35.72%
   5, K, 5, 2, 2, 1, 42.86%
   ```

   There are three '**A**' rooms and five weekdays available, so the third field is 15. The fifth field then is 3 × 2 = 6. There were two successful bookings for the 'A' room, booking **#3** and **#11**. Booking request **#3** booked all seven days = 5 weekdays and 2 days for the

weekend; #11 booked only the weekend. Obviously, **#3** and **#11** will be assigned separate rooms; but we are **not interested** in the room itself (hence, there are no room numbers for each of the three 'A' rooms). So the fourth field for '**A**' room has `5`, due to booking **#11**; and sixth field has `2 + 2 = 4`. (#3 booked Saturday and Sunday and so did **#11**).

Note that your program may not generate exactly "`42.86%`" for the `<%occ_1>`. We allow for `± 0.05` margin of error.

Note that since the rooms.txt file lists the room codes in this order: 'A', 'B', 'C', 'D', 'K'; then the `report.txt` file also lists the lines in the same order.

I. Deliverables

1. Check the syllabus for machine project due date. Submission Time stamp should not go beyond **11:59pm**.
2. You are to submit and email your source code (.C) to projectssubmission@gmail.com and another copy must be uploaded in Canvas. Appropriate section in Canvas for this purpose will be created later on.
3. It is expected in your .C program that you incorporate an in-text documentation or internal documentation. This means that details of the program are explained by comments within the code. Your internal documentation should at the minimum include the following:
    a. A "block comment" should be placed at the head of every major/crucial function or subroutines. This will include:
        i. The function name
        ii. Purpose of the function
        iii. Function's pre- and post- conditions
        iv. Function's return value (if any), and
        v. A list of all parameters including direction of information transfer (i.e. into this function, out from the function back to the calling function or both) and their purposes.
    b. Each variable, constant, files, structures and the like must have a brief comment next to its declaration that explains its purpose. This applies as well to all fields of structure declarations.
    c. Complex sections of codes and any other parts of the program that need some explanation should have comments just ahead of them or embedded in them.
4. Right after the very last close curly bracket (}) in your main() function, you are to document as comments (within the boundaries of /* */) the different test data and test cases you used to test the correctness of your program. If sample text files were created as part of your test cases, submit them as well along with your C code.

***** End of MP specification *****