**CPSC 2150: Week 4 Homework Bitwise Operators, Priority Queues**

**Due: As indicated by submission link**

**Total Marks: 10**

**Instructions – PLEASE READ**

1. This work is to be done individually.
2. You should submit only one version via D2L. Check instructions from TA regarding what to submit (zip/just code). Code files must always be included.
3. Keep a copy of everything you submit in some online storage that is accessible by you only.

*Copying another student’s work or submitting anything that is not your own work will result in a zero for all parties involved. Academic alert may follow!*

# EXERCISES

1. A hotel wants to manage the booking status of their 60 rooms (room number 1 to room number 60). Here are the tasks you need to complete:
   * Create a class named Hotel with a private data member named **rooms**. Type is to be determined by you. It should be able to hold booking status information for 60 rooms
   * Add a default constructor that initializes **rooms** to 0 (i.e. initially, none of the rooms are booked (0 for a room's bit means it is is not booked).
   * Add a method **bool CheckAvailability(int roomNumber)** that validates the roomNumber (should be between 1 and 60; otherwise output a suitable message and return false), then checks to see if room has already been reserved *using bitwise operators* – return true if room is available, false if room has already been reserved.
   * Add a method **void BookRoom(int roomNumber)** that checks to see if roomNumber is available; if yes then roomNumber is reserved *using bitwise operators* (set the bit for that roomNumber in **rooms** to 1 indicating that it has now been reserved). Output appropriate messages to the console.
   * Add a method **void CheckOut(int roomNumber)** that checks to see if roomNumber is has been reserved already; if yes then roomNumber is made available for booking again *using bitwise operators* (reset the bit for that roomNumber in **rooms** to 0 indicating that it is now available for booking again). Output appropriate messages to the console.
   * Add a method **void ShowRooms()** that displays each room's number and its current reservation status (i.e. reserved or available). Any loop used in this method must be controlled *using bitwise operators*.
     1. This method should also output the total number of rooms that are reserved and the total number of rooms that are available – both values must be calculated *using bitwise operators*.
   * Test all features in HotelDriver.cpp by implementing a recursive function named **Menu** that can be used to do the following:
     1. **0**: Exit Program (this will be the only way to successfully end the program)
     2. **1**: Show Rooms (displays the current booking status of all rooms).
     3. **2**: Book Room (allows the user to input a room number that they would like to reserve and reserve it if it is a valid number and room is available).
     4. **3**: Check Out (allows the user to input a room number that they would like to check out of and make that room available if it is a valid number and is currently reserved).
2. Implementing a Priority Queue using a Skip List
   * For this problem you will need to have the SkipList class implemented with insert, delete, search methods. If any solution is posted (by me or the marker) related to SkipLists then feel free to use it. Feel free to add (or change or remove) members to the SkipList class. Just ensure it remains a skip list data structure.
   * “… In [computer science](https://en.wikipedia.org/wiki/Computer_science), a **priority queue** is an [abstract data-type](https://en.wikipedia.org/wiki/Abstract_data_type) similar to a regular [queue](https://en.wikipedia.org/wiki/Queue_(abstract_data_type)) or [stack](https://en.wikipedia.org/wiki/Stack_(abstract_data_type)) data structure. Each element in a priority queue has an associated *priority.* In a priority queue, elements with high priority are served before elements with low priority. …” - wikipedia
   * The priority queue holds data in a skip list where each node is doubly linked at every level
     1. Each node in the skip list contains an integer data, a non-negative integer priority (this will also be the level index of the node in the skiplist once inserted, default 0 because every node must be part of level 0), a vector of next links and a vector of previous links where the number of links in each vector determines how many levels the node is part of.
        1. Maximum value for priority is the maximum allowed level for the skip list, minimum value for priority is 0
   * Add methods Enqueue, Dequeue, Process, Show to the Priority Queue class
   * When new item is being enqueued set its data and priority to random integers (priority is a random integer between 0 and MAXIMUM\_ALLOWED\_LEVEL\_INDEX which is a property of the SkipList class)
   * Priority of a node determines its level in the skiplist (no coin tosses)
   * When a priority queue is processed, all higher priority items are processed before lower priority items
     1. Once an item is processed it is Dequeued from the PriorityQueue (i.e. removed from the SkipList)
     2. The Process method must output the data values in the order they were processed
   * Display the Priority Queue (i.e. the skip list from highest level to lowest level, each level on a separate line, show each node’s data value) to check your code
   * Test all features in a main function.