Assignment #2

Part C:

For room, because I know they size of room. I chose array to store rooms in partB. Array is easier for me to implement and I can directly access to each room based on index.
For customer, I used vector to store it. If I choose LinkedList, it is not space efficient. Vector can automatically grows so I used vector.

Part D: As what I said in part c, using array for room is convienient to directly access each room in the array.
Using vector for customers is not more efficient than using array but is convienient to add element if you do not know the size of input. If I use LinkedList for customers, it seems like it is a waste of memory.

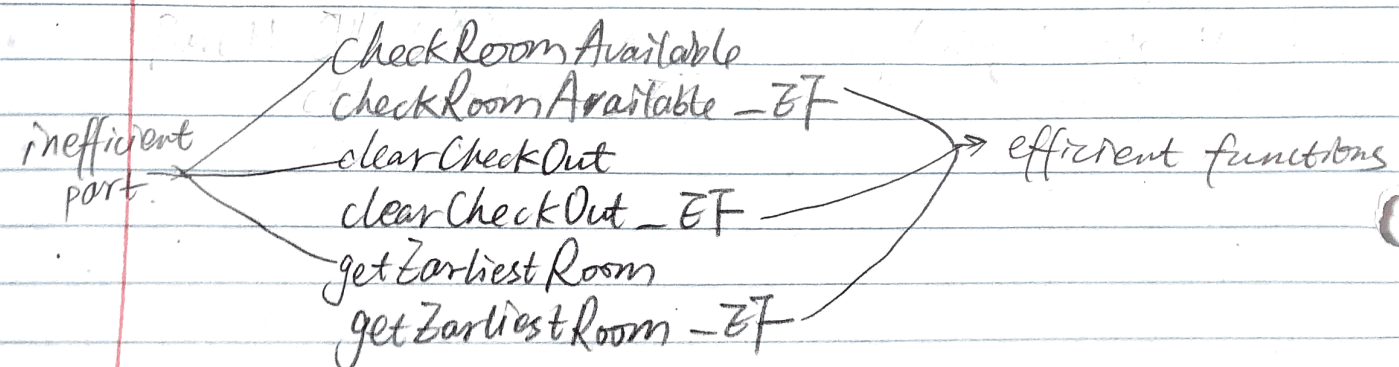
Part E: if I have more rooms, I will not use array. Thus, I use LinkedList to store room.
To speed up the process of answering customers, I want to seperate available rooms and unavailable rooms from room lists.
First of all, I used three LinkedLists for rooms with 1 bed, 2 beds, and 3 beds.

Part F: Yes, self-organizing techniques are useful. For my room lists, I want to move the unavailable room to the end of list after serving each customer. This, available rooms are always at front. It will take less time because every time I am finding an empty I do not need to go through each room.

To make it more efficient, I made three room lists that store available rooms and three unavailable room lists.
Every time I give a customer a room, I remove a room from available list and add it to unavailable list. Unavailable room list is actually a priority queue. When I add a room in the queue, it will be ordered by the check out date of each room. The earliest room that will be checked out will be arranged at front. Thus, it is much more efficient.

① when I look for an available room, I do not need to traverse available room list, I can simply sign the first room in the available list to the customer. (when the available list is not empty)

② Before I look for room in available room list, I will remove the rooms that have already checked out in unavailable room list to available room list. Thus, when the available room list is empty and I need to find out which is the earliest room, the first one in unavailable room list will be the one I want so I do not need to go through each room in the unavailable room list.



Part H: the difference of time complexity is significant.
For inefficient part, it takes around 50 seconds to arrange every customers in the file.
The efficient functions only take 10 seconds.
I also used a integer — "count" to count how many times my functions visit a room.

∴ "count" for inefficient functions is 48782293
"count" for efficient functions is 601532