Group Members: Burayag, Ethan Axl S. - Del Rosario, Ezra Jeonadab G.
Group #: 4

MCO1 - Test Script
CCPROG3

Class : ReservationSystem						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
create Hotel	1	Create a hotel with a unique name	Input: “SOGO”	hotelName : “SOGO”	hotelName : “SOGO”	P
	2	Create a hotel with a duplicate name	Input: “SOGO” hotelNames : “SOGO”, “Hotel1”	Prompt: *Hotel Name Already Exists*	Prompt: *Hotel Name Already Exists*	P
	3	Create a hotel with a duplicate name but have different lettercases.	Input: “sogo” hotelNames: “SOGO”, “Hotel1”,	hotelName : “sogo”	hotelName : “sogo”	P
	4	roomNum is less than 1	roomNum = 0	Prompt: *Invalid Number of Rooms*	Prompt: *Invalid Number of Rooms*	P
	5	roomNum is greater than 50	roomNum = 51	Prompt: *Invalid Number of Rooms*	Prompt: *Invalid Number of Rooms*	P
	6	roomNum is valid	roomNum = 5	hotelName : “SOGO” roomNum : 5	hotelName : “SOGO” roomNum : 5	P
viewHotel	1	Enter a valid input to any input field for viewing.	*Assume supposed integer input* Enter the number of the command to run: “1”	InputReceived : 1	InputReceived : 1	P
	2	Enter an invalid input to any input field for viewing	*Assume supposed integer input* Enter the number of the command to	Prompt: *Invalid Integer Input*	Prompt: *Invalid Integer Input*	P

			run: “one”			
	3	Prematurely exit/quit out of the view	*Assume supposed integer input* Enter the number of the command to run: “Quit”	*Exit view method*	*Exit view method*	P
	4	Prematurely exit/quit out of the view (Input with varying lettercase)	*Assume supposed integer input* Enter the number of the command to run: “qUiT”	*Exit view method*	*Exit view method*	P
rename Hotel	1	Rename a hotel with a unique name	hotelToRename: “Sogo” newName: “SOFITEL” hotelNames : “SOFITEL” “Heritage:	hotelName : “SOFITEL”	hotelName : “SOFITEL”	P
	2	Rename a hotel with an existing name	hotelToRename: “Sogo” newName: “SOFITEL” hotelNames : “SOFITEL” “Heritage:	Prompt: *Hotel Name Already Exists*	Prompt: *Hotel Name Already Exists*	P
	3	Renaming a hotel with its original name.	hotelToRename: “Sogo” newName: “Sogo” hotelNames : “SOFITEL” “Heritage:	Prompt: *Hotel Name Already Exists*	Prompt: *Hotel Name Already Exists*	P
remove Hotel	1	The index given is valid.	hotelToDelete : 0 hotelList : “SOGO”	hotelList : “SOFITEL”	hotelList : “SOFITEL”	P

			“SOFITEL”			
	2	The index given is invalid	hotelToDelete : -1 hotelList : “SOGO” “SOFITEL”	Prompt : *Input Invalid*	Prompt : *Input Invalid*	P
	3	Confirm Modification	hotelToDelete : 1 Confirm Remove? : “Yes” hotelList : “SOGO” “SOFITEL”	hotelList : “SOGO”	hotelList : “SOGO”	P
	4	Reject Modification	hotelToDelete : 1 Confirm Remove? : “No” hotelList : “SOGO” “SOFITEL”	hotelList : “SOGO” “SOFITEL”	hotelList : “SOGO” “SOFITEL”	P
manage Hotel	1	Run renameHotel method with valid inputs	Input : “1”	*run renameHotel()*	*run renameHotel()*	P
	2	Run addRoom method from the Hotel class with valid inputs	Input : “2”	*run addRoom()*	*run addRoom()*	P
	3	Run removeRoom method from the Hotel class with valid inputs	Input : “3”	*run removeRoom()*	*run removeRoom()*	P
	4	Run updateBasePrice method from the Hotel class with valid inputs	Input : “4”	*run updateBasePrice()*	*run updateBasePrice()*	P
	5	Run removeReservation method from the Hotel class with valid inputs	Input : “5”	*run removeReservation()*	*run removeReservation()*	P
	6	Run removeHotel method	Input : “6”	*run removeHotel()*	*run removeHotel()*	P
	7	Number of rooms to add is invalid (addRoom)	roomNum : “51”	Prompt: *Invalid Input*	Prompt: *Invalid Input*	P

	8	Number of rooms to delete is invalid (removeRoom)	roomNum : “51”	Prompt: *Invalid Input*	Prompt: *Invalid Input*	P
	9	Updating base price when reservations exist.	Input : “4”	Prompt: *Existing reservations*	Prompt: *Existing reservations*	P
	10	Remove reservations on an empty reservation list	Input: “5”	Prompt: *No Existing reservations*	Prompt: *No Existing reservations*	P
	11	Quit	Input: “Quit”	*Exit to Menu*	*Exit to Menu*	P
simulate Booking	1	The check-in date is set as 31	checkInDate : 31	Prompt: *No Room Available for date*	Prompt: *No Room Available for date*	P
	2	The check-out date is set as 1	checkOutDate: 1	Prompt: *No Room Available for date*	Prompt: *No Room Available for date*	P
	3	The dates are valid	checkInDate : 1 checkOutDate : 2	*Reservation Added to the Reservation List*	*Reservation Added to the Reservation List*	P
isExisting	1	Hotel name is non-existent	Input: “Jollibee” hotelList : “SOGO” “SOFITEL”	Result = -1	Result = -1	P
	2	Hotel name exists	Input: “SOGO” hotelList : “SOGO” “SOFITEL”	Result = 0	Result = 0	P
	3	Similar hotel names exist but with different lettercases	Input: “SOGO” hotelList : “soGo” “SOGO”	Result = 1	Result = 1	P
run System	1	Run createHotel method	Input: “1”	*run createHotel()*	*run createHotel()*	P
	2	Run viewHotel method	Input: “2”	*run viewHotel()*	*run viewHotel()*	P
	3	Run manageHotel method	Input: “3”	*run manageHotel()*	*run manageHotel()**	P

	4	Run simulateBooking method	Input: “4”	*run simulateBooking ()*	*run simulateBooking ()*	P
	5	Quit	Input: “Quit”	*Exit to Menu*	*Exit to Menu*	P
	6	Hotel name is existing (createHotel)	Input: “SOGO” hotelNames : “SOGO”, “Hotel1”	Prompt: *Hotel Name Already Exists*	Prompt: *Hotel Name Already Exists*	P
	7	Hotel list is empty (viewHotel, manageHotel, simulateBooking)	Input: “SOGO” hotelNames :	Prompt: *Hotel list is empty*	Prompt: *Hotel list is empty*	P

Class : Hotel						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
sortRoom List	1	Sorts room names that are in increasing order already based on room numbers	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”}	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”}	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”}	P
	2	Sorts room names that are in decreasing order based on room numbers	roomListNames: {“A5”, “A4”, “A3”, “A2”, “A1”}	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”}	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”}	P
	3	Sorts room names that are in no particular order based on room numbers	roomListNames: {“A10”, “A7”, “A5”, “A4”, “A1”}	roomListNames: {“A1”, “A4”, “A5”, “A7”, “A10”}	roomListNames: {“A1”, “A4”, “A5”, “A7”, “A10”}	P
check Availability	1	Checks availability of a room with no existing reservations for the entire month	Room “A1” Reservation: None Check Availability: checkIn: 6 checkOut: 10	True	True	P
	2	Checks availability of a room for a different range of dates from an existing reservation	Room “A1” Reservation 1: checkIn:1 checkOut:5 Check Availability: checkIn: 6 checkOut: 10	True	True	P
	3	Checks availability of a room for a similar range of dates from an existing reservation	Room “A1” Reservation 1: checkIn:1 checkOut:5 Check Availability: checkIn: 3 checkOut: 5	False	False	P
	4	Checks availability of a room for an overlap range of dates from an existing reservation	Room “A1” Reservation 1: checkIn:1 checkOut:5	False	False	P

			Check Availability: checkIn: 4 checkOut: 10			
checkAll Room Availability	1	Checks all room availability of the hotel with no existing reservations	reservedRooms List: None	True	True	P
	2	Checks all room availability of the hotel with an existing reservation	reservedRooms List: {“A1”}	False	False	P
	3	Checks all room availability of the hotel with multiple existing reservations	reservedRooms List: {“A1”, “A2”, “A3”}	False	False	P
getRoom Index	1	Gets the index of an existing room name from the room list	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”} roomName: “A2”	1	1	P
	2	Gets the index of a non-existing room name from the room list	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”} roomName: “A6”	-1	-1	P
	3	Gets the index of an existing room name found at the end of the room list	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”} roomName: “A5”	4	4	P
addRoom	1	Creates room names that are sequential from the existing complete room list and creates room objects and adds to the room list (Confirmed Modification)	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”} numRoom: 5 Confirm Modification: Yes	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”, “A6”, ”A7”, ”A8”, “A9”, “A10”}	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”, “A6”, ”A7”, ”A8”, “A9”, “A10”}	P
	2	Creates room names that	roomListNames:	roomListNames:	roomListNames:	P

		are sequential from the existing broken room list and creates room objects and adds to the room list (Confirmed Modification)	{“A1”, “A3”, “A4”, “A6”, “A7”} numRoom: 5 Confirm Modification: Yes	{“A1”, “A2”, “A3”, “A4”, “A5”, “A6”, “A7”, “A8”, “A9”, “A10”}	{“A1”, “A2”, “A3”, “A4”, “A5”, “A6”, “A7”, “A8”, “A9”, “A10”}	
	3	Creates room names that are sequential from the existing complete room list but does not create room objects and does not add to the room list (Rejected Modification)	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”} numRoom: 5 Confirm Modification: No	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”}	roomListNames: {“A1”, “A2”, “A3”, “A4”, “A5”}	P
remove Room	1	Checks validity of list of index (Valid) and availability of list of concerned rooms (All does not have any reservation) and removes the rooms from the list (Confirm Modification)	roomList - Availability: {Room1 - True, Room2 - True, Room3 - True, Room4 - True, Room5 - True} index: {2,4} Confirm Modification: Yes	True roomList - Availability: {Room1 - True, Room2 - True, Room4 - True}	True roomList - Availability: {Room1 - True, Room2 - True, Room4 - True}	P
	2	Checks validity of list of index (Valid) and availability of list of concerned rooms (All does not have any reservation) but does not remove the rooms from the list (Reject Modification)	roomList - Availability: {Room1 - True, Room2 - True, Room3 - True, Room4 - True, Room5 - True} index: {2,4} Confirm Modification: No	True roomList - Availability: {Room1 - True, Room2 - True, Room3 - True, Room4 - True, Room5 - True}	True roomList - Availability: {Room1 - True, Room2 - True, Room3 - True, Room4 - True, Room5 - True}	P
	3	Checks validity of list of index (Valid) and availability of list of concerned rooms (One has	roomList - Availability: {Room1 - True, Room2 - True,	False	False	P

		existing reservation)	Room3 - False, Room4 - True, Room5 - True} index: {2,4}			
	4	Checks validity of list of index (Invalid)	roomList - Availability: {Room1 - True, Room2 - True, Room3 - False, Room4 - True, Room5 - True} index: {2,8}	False	False	P
update RoomPrice	1	Checks validity of new price value (Valid) and updates the room price (Confirm Modification)	price: 1299 newPrice: 1000 Confirm Modification: Yes	True price: 1000	True price: 1000	P
	2	Checks validity of new price value (Valid) but does not update the room price (Reject Modification)	price: 1299 newPrice: 1000 Confirm Modification: No	True price: 1299	True price: 1299	P
	3	Checks validity of new price value (Invalid)	price: 1299 newPrice: 50	False	False	P
add Reservation	1	Checks validity of date values (Valid) and availability of a room (There exists an available room) and adds a reservation to the reservation list (Confirm Modification)	reservationList: {Reservation1, Reservation2, Reservation3} guestName: “A” checkInDate: 1 checkOutDate: 5 availableRoom: Room1 Confirm Modification: Yes	True reservationList: {Reservation1, Reservation2, Reservation3, Reservation4}	True reservationList: {Reservation1, Reservation2, Reservation3, Reservation4}	P
	2	Checks validity of date values (Valid) and availability of a room (There exists an available room) but does not add a reservation to the	reservationList: {Reservation1, Reservation2, Reservation3} guestName: “A”	True reservationList: {Reservation1, Reservation2, Reservation3}	True reservationList: {Reservation1, Reservation2, Reservation3}	P

		reservation list (Reject Modification)	checkInDate: 1 checkOutDate: 5 availableRoom: Room1 Confirm Modification: No			
	3	Checks validity of date values (Valid) and availability of a room (There is no available room)	reservationList: {Reservation1, Reservation2, Reservation3} guestName: “A” checkInDate: 1 checkOutDate: 5 availableRoom: None	False	False	P
	4	Checks validity of date values (Invalid)	reservationList: {Reservation1, Reservation2, Reservation3} guestName: “A” checkInDate: 8 checkOutDate: 5 availableRoom: None	False	False	P
remove Reservation	1	Checks validity of index value (Valid) and removes the reservation from the reservation list (Confirmed Modification)	index: 4 reservationList: {Reservation1, Reservation2, Reservation3, Reservation4, Reservation5} Confirm Modification: Yes	True reservationList: {Reservation1, Reservation2, Reservation3, Reservation4}	True reservationList: {Reservation1, Reservation2, Reservation3, Reservation4}	P
	2	Checks validity of index value (Valid) but does not remove the reservation from the reservation list (Rejected Modification)	index: 4 reservationList: {Reservation1, Reservation2, Reservation3, Reservation4, Reservation5}	True reservationList: {Reservation1, Reservation2, Reservation3, Reservation4, Reservation5}	True reservationList: {Reservation1, Reservation2, Reservation3, Reservation4, Reservation5}	P

			Confirm Modification: No			
	3	Checks validity of index value (Invalid)	index: 10 reservationList: {Reservation1, Reservation2, Reservation3, Reservation4, Reservation5}	False	False	P
getTotal Available Rooms	1	Checks validity of date value (Valid) and returns the total number of available rooms	date: 10 roomList - Availability: {Room1 - True, Room2 - True, Room3 - False, Room4 - False, Room5 - False}	2	2	P
	2	Checks validity of date value (Valid but Max Date) *Checking-in not available for Max Date	date: 31 roomList - Availability: {Room1 - True, Room2 - True, Room3 - False, Room4 - False, Room5 - False}	-1	-1	P
	3	Checks validity of date value (Invalid)	date: 0 roomList - Availability: {Room1 - True, Room2 - True, Room3 - False, Room4 - False, Room5 - False}	-1	-1	P
getTotal Booked Rooms	1	Checks validity of date value (Valid) and returns the total number of booked rooms	date: 10 roomList - Availability: {Room1 - True, Room2 - True, Room3 - False, Room4 - False, Room5 - False}	3	3	P

	2	Checks validity of date value (Invalid - Greater than max)	date: 32 roomList - Availability: {Room1 - True, Room2 - True, Room3 - False, Room4 - False, Room5 - False}	-1	-1	P
	3	Checks validity of date value (Invalid - Less than max)	date: 0 roomList - Availability: {Room1 - True, Room2 - True, Room3 - False, Room4 - False, Room5 - False}	-1	-1	P
getRoom Info	1	Checks validity of index value (Valid) and returns the corresponding room info	index: 2 roomList: {Room1, Room2, Room3, Room4, Room5}	Room Info of Room3	Room Info of Room3	P
	2	Checks validity of index value (Invalid - Greater than size)	index: 8 roomList: {Room1, Room2, Room3, Room4, Room5}	null	null	P
	3	Checks validity of index value (Invalid - Less than size)	index: -1 roomList: {Room1, Room2, Room3, Room4, Room5}	null	null	P
get Reservation Info	1	Checks validity of index value (Valid) and returns the corresponding reservation info	index: 2 reservationList: {Reservation1, Reservation2, Reservation3, Reservation4, Reservation5}	Reservation Info of Reservation3	Reservation Info of Reservation3	P
	2	Checks validity of index value (Invalid - Greater than size)	index: 6 reservationList: {Reservation1, Reservation2,	null	null	P

			Reservation3, Reservation4, Reservation5}			
	3	Checks validity of index value (Invalid - Less than size)	index: 0 reservationList: {Reservation1, Reservation2, Reservation3, Reservation4, Reservation5}	null	null	P

Class : Reservation
No significant methods to test for system logic

Class : Room						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
set Availability	1	endDate < 31 (MAX_DATE)	Input : setAvailability(1, 30, false)	availability = false, false,..., true	availability = false, false,..., true,	P
	2	endDate == 31	Input : setAvailability(1, 31, false)	availability = false, false,..., false	availability = false, false,..., false	P
	3	No significant test case to be conducted				