COMPUTER SCIENCE A
SECTION II
Time – 1 hour and 45 minutes
Number of questions – 4

Directions: SHOW ALL YOUR WORK REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Quick Reference found in the Appendix have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.

1. A restaurant is working to implement a new system for reservations to their restaurant. A reservation specifies the name of the reserver and the number of people in the party. The declaration of the Reservation class is shown below:

```
public class Reservation
{
        /** Constructs a new Reservation object, with name set to lowercase. */
        public Reservation(String name, int seats)
        { /* implementation not shown */ }

        /** @return the name of the reserver
        public String getName()
        { /* implementation not shown */ }

        /** @return the number of seats in a party
        public int getSeats()
        { /* implementation not shown */ }

        // There may be instance variables, constructors, and methods that are not shown.
}
```

The Restaurant class maintains a list of the reservations in order from first to last. The declaration of the Restaurant class is shown below.

```
public class Restaurant
{
	/** The list of all reservations*/
	private ArrayList<Reservation> tableList;

	/** The current index at which ones reservation is located*/
	private int currentPlace;

	/** Constructs a new Restaurant object */
	public Restaurant()
	{
		tableList = new ArrayList<Reservation>();
		currentPlace = 0;
	}

	/** @returns the index, place, at which reservation is located.
	 * If it cannot find the reservation it will return a -1.
	 * Make sure to turn the name into lowercase to prevent any case sensitive errors.
	 * @param the name of the reserver being located & the party size.
	 */
	public int findReservation(String name, int seats)
	{ /* to be implemented in part (a) */ }

	/** @return the updated list, (do NOT create a new list or duplicate)
	 * A person will check in for a reservation, if the person has a reservation
	 * he will be removed from the list. If he is not on the list add him to the list.
	 * Make your name into lower case.
	 * Make sure to use part a method findReservation(String name, int seats)
	 * @paran the name of the reserver being located & the party size.
	 */
	public ArrayList<Reservation> newListUpdated(String name, int seats)
	{ /* to be implemented in part (b) */ }

	/** Adds reservation r to the list*/
	public void add(Reservation r)
	{ tableList.add(r); }
```

```
    /** removes reservation located at index, i, in the list*/
    public void remove(int i)
    { tableList.remove(i); }

    /** @return the currentPlace*/
    public int getCurrentPlace()
    { return currentPlace; }

    // There may be instance variables, constructors, and methods that are not shown.
}
```

(a) The findReservation method computes and returns the place at which a reservation is located in the list, tableList. If the reservation is not located in the list it will return -1. Make sure that your method does not neglect cases in which the ArrayList is empty.

For example, consider the following code segment.

```
Restaurant javaCoffee =  new Restaurant();
javaCoffee.add(new Reservation("carlos", 5));
javaCoffee.add(new Reservation("compsci", 9));
```

After the code segment has been executed, the contents of the master order are shown as follows.

“carlos” 5	“compsci” 9

The method call javaCoffee.findReservation(“compsci”, 9) returns the index at which the reservation is located.

It would return: **1**

Followed by the method call javaCoffee.findReservation(“matthew”, 4) returns the index at which the reservation is located, in this case it is not located in the restaurant list.

It would return: **-1**

Complete the method findReservation below:

```
    /** @returns the index, place, at which reservation is located.
```

```
 * If it cannot find the reservation it will return a -1.
 * Make sure to turn the name into lowercase to prevent any case sensitive errors.
 * @param the name of the reserver being located & the party size.
 */
public int findReservation(String name, int seats)
```

(b) The newListUpdated method will update the restaurant when a reservation has been searched. If the reservation being searched has already been reserved you will remove the reservation from the restaurant. Otherwise if the reservation being searched is not on the restaurant's list, it will update the list by adding it to the end of the list. ONLY for part b, assume that all of the reservations contain a name and a number and are not null.

For example, consider the following code segment.

```
Restaurant javaCoffee =  new Restaurant();
javaCoffee.add(new Reservation("carlos", 5));
javaCoffee.add(new Reservation("compsci", 9));
```

After the code segment has been executed, the contents of the master order are shown as follows.

"carlos" 5	"compsci" 9

The method call javaCoffee.newListUpdated("carlos", 5) returns the updated version as follows.

compsci" 9

Followed by the method call javaCoffee.newListUpdated("matthew", 4) returns the updated version as follows.

"compsci" 9	"matthew" 4

Complete the method findReservation below:

```
/** @return the updated list, (do NOT create a new list or duplicate)
 * A person will check in for a reservation, if the person has a reservation
 * he will be removed from the list. If he is not on the list add him to the list.
 * Make your name into lower case.
 * Make sure to use part a method findReservation(String name, int seats)
 * @paran the name of the reserver being located & the party size.
 */
public ArrayList<Reservation> newListUpdated(String name, int seats)
```