Name _____ Date _____

COMPUTER SCIENCE A

SECTION II

Time – 23 minutes

Number of questions – 1

brought to you by Carlos V ('23)

**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 customer 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 customer
  public String getName()
  { /* implementation not shown */ }

  /** @return the size of the 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> waitingList;

/** @returns the index, where reservation is located.
 * If it cannot find the reservation, it will return -1.
 * @param the name of the customer and their party size.
 *
 * Precondition: name is in lower case
*  or one more reservation.
*/
public int findReservation(String name, int seats)
   { /* to be implemented in part (a) */ }

/** checkIn will update the waitingList
 * If the reservation is found, it will be removed from waitingList.
 * If it is not found, a new Reservation will be added to waitingList.
 * Make sure not to duplicate code.  You made presume that the method
 * written in part(a) works perfectly.
 * @param the name of the customer being located the party size.
 * @return true if the reservation was found, and removed from the waitingList
 *         false if it was not found, and added to the waitingList
 * Precondition: name is in lower case, seats >=0
 * Postcondition: tables with either have one less reservation,
 *                or one more reservation
 */

public boolean checkIn (String name, int seats)
{ /* to be implemented in part (b) */ }

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

(a) The `findReservation` method of the `Restaurant` class computes and returns the place at which a reservation is located in the waiting list. If the reservation is found, it should how many parties are ahead, otherwise, if it is not found, it should return -1.

For example, if there were a `Restaurant` instance called `javaCafe` which has the following two reservations:

| | |
|---|---|
| "carlos"<br>5 | "compsci"<br>9 |

the method call `javaCafe.findReservation("compsci", 9)` would return 1, the index where `compsci` occurs in the waiting list, as well as the number of parties that are ahead of `compsci`.

The method call `javaCoffee.findReservation("bruno", 4)` returns -1, since it is not found in the waiting list.

Complete the `Restaurant` method `findReservation` below:

```
public int findReservation(String name, int seats)
```

(b) The `Restaurant` method `checkIn` will remove the reservation if it is on the waiting list, otherwise it will add a new `Reservation` to the waiting list. If the reservation was found, it should return `true`, otherwise it should return `false`. Avoid duplicating code. You may presume the method described in part (a) works properly.

For example, if there were a `Restaurant` instance called `javaCafe` which has two entries:

| | |
|---|---|
| "carlos"<br>5 | "compsci"<br>9 |

The method call `javaCafe.checkIn("carlos", 5)` is in the waiting list, and so it is removed, and a `true` is returned, leaving the waiting list with:

| |
|---|
| compsci"<br>9 |

On the other hand, the call `javaCafe.checkIn("bruno", 1)` is not in the waiting list, so it is added to the waiting list, and a false is returned, leaving the waiting list with:

| | | |
|---|---|---|
| "carlos"<br>5 | "compsci"<br>9 | "bruno"<br>1 |

Complete the `Restaurant` method `checkIn` below:

```
public boolean checkIn (String name, int seats)
```