

2008 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

COMPUTER SCIENCE A SECTION II

Time—1 hour and 45 minutes

Number of questions—4

Percent of total grade—50

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 travel agency maintains a list of information about airline flights. Flight information includes a departure time and an arrival time. You may assume that the two times occur on the same day. These times are represented by objects of the `Time` class.

The declaration for the `Time` class is shown below. It includes a method `minutesUntil` that returns the difference (in minutes) between the current `Time` object and another `Time` object.

```
public class Time
{
    /** @return difference, in minutes, between this time and other;
     *     difference is negative if other is earlier than this time
     */
    public int minutesUntil(Time other)
    { /* implementation not shown */ }

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

For example, assume that `t1` and `t2` are `Time` objects where `t1` represents 1:00 P.M. and `t2` represents 2:15 P.M. The call `t1.minutesUntil(t2)` will return 75 and the call `t2.minutesUntil(t1)` will return -75.

2008 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

The declaration for the `Flight` class is shown below. It has methods to access the departure time and the arrival time of a flight. You may assume that the departure time of a flight is earlier than its arrival time.

```
public class Flight
{
    /** @return time at which the flight departs
     */
    public Time getDepartureTime()
    { /* implementation not shown */ }

    /** @return time at which the flight arrives
     */
    public Time getArrivalTime()
    { /* implementation not shown */ }

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

2008 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

A trip consists of a sequence of flights and is represented by the `Trip` class. The `Trip` class contains an `ArrayList` of `Flight` objects that are stored in chronological order. You may assume that for each flight after the first flight in the list, the departure time of the flight is later than the arrival time of the preceding flight in the list. A partial declaration of the `Trip` class is shown below. You will write two methods for the `Trip` class.

```
public class Trip
{
    private ArrayList<Flight> flights;
        // stores the flights (if any) in chronological order

    /** @return the number of minutes from the departure of the first flight to the arrival
     *         of the last flight if there are one or more flights in the trip;
     *         0, if there are no flights in the trip
     */
    public int getDuration()
    {    /* to be implemented in part (a) */    }

    /** Precondition: the departure time for each flight is later than the arrival time of its
     *         preceding flight
     * @return the smallest number of minutes between the arrival of a flight and the departure
     *         of the flight immediately after it, if there are two or more flights in the trip;
     *         -1, if there are fewer than two flights in the trip
     */
    public int getShortestLayover()
    {    /* to be implemented in part (b) */    }

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

2008 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Complete method
- `getDuration`
- below.

```

/** @return the number of minutes from the departure of the first flight to the arrival
 *         of the last flight if there are one or more flights in the trip;
 *         0, if there are no flights in the trip
 */
public int getDuration()

```

- (b) Write the
- `Trip`
- method
- `getShortestLayover`
- . A layover is the number of minutes from the arrival of one flight in a trip to the departure of the flight immediately after it. If there are two or more flights in the trip, the method should return the shortest layover of the trip; otherwise, it should return -1.

For example, assume that the instance variable `flights` of a `Trip` object `vacation` contains the following flight information.

	Departure Time	Arrival Time	Layover (minutes)
Flight 0	11:30 a.m.	12:15 p.m.	} 60
Flight 1	1:15 p.m.	3:45 p.m.	
Flight 2	4:00 p.m.	6:45 p.m.	} 15
Flight 3	10:15 p.m.	11:00 p.m.	
			} 210

The call `vacation.getShortestLayover()` should return 15.

Complete method `getShortestLayover` below.

```

/** Precondition: the departure time for each flight is later than the arrival time of its
 *         preceding flight
 * @return the smallest number of minutes between the arrival of a flight and the departure
 *         of the flight immediately after it, if there are two or more flights in the trip;
 *         -1, if there are fewer than two flights in the trip
 */
public int getShortestLayover()

```