

AP® Computer Science A
2008 Free-Response Questions

The College Board: Connecting Students to College Success

The College Board is a not-for-profit membership association whose mission is to connect students to college success and opportunity. Founded in 1900, the association is composed of more than 5,000 schools, colleges, universities, and other educational organizations. Each year, the College Board serves seven million students and their parents, 23,000 high schools, and 3,500 colleges through major programs and services in college admissions, guidance, assessment, financial aid, enrollment, and teaching and learning. Among its best-known programs are the SAT®, the PSAT/NMSQT®, and the Advanced Placement Program® (AP®). The College Board is committed to the principles of excellence and equity, and that commitment is embodied in all of its programs, services, activities, and concerns.

© 2008 The College Board. All rights reserved. College Board, Advanced Placement Program, AP, AP Central, SAT, and the acorn logo are registered trademarks of the College Board. PSAT/NMSQT is a registered trademark of the College Board and National Merit Scholarship Corporation.
Permission to use copyrighted College Board materials may be requested online at:
www.collegeboard.com/inquiry/cbpermit.html.

Visit the College Board on the Web: www.collegeboard.com.
AP Central is the official online home for the AP Program: apcentral.collegeboard.com.

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 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

3. This question involves reasoning about the code from the GridWorld case study. A copy of the code is provided as part of this exam.

   An opossum is an animal whose defense is to pretend to be dead. The `OpossumCritter` class, shown below, will be used to represent the opossum in the grid. An `OpossumCritter` classifies its neighbors as friends, foes, or neither. It is possible that a neighbor is neither a friend nor a foe; however, no neighbor is both a friend and a foe. If the `OpossumCritter` has more foes than friends surrounding it, it will simulate playing dead by changing its color to black and remaining in the same location. Otherwise, it will behave like a `Critter`. If the `OpossumCritter` plays dead for three consecutive steps, it is removed from the grid.

   You will implement two of the methods in the following `OpossumCritter` class.

```
public class OpossumCritter extends Critter
{
  private int numStepsDead;

  public OpossumCritter()
  {
    numStepsDead = 0;
    setColor(Color.ORANGE);
  }

  /** Whenever actors contains more foes than friends, this OpossumCritter plays dead.
   *  Postcondition: (1) The state of all actors in the grid other than this critter and the
   *  elements of actors is unchanged. (2) The location of this critter is unchanged.
   *  @param actors a group of actors to be processed
   */
  public void processActors(ArrayList<Actor> actors)
  {  /* to be implemented in part (a) */  }


  /** Selects the location for the next move.
   *  Postcondition: (1) The returned location is an element of locs, this critter's current location,
   *  or null. (2) The state of all actors is unchanged.
   *  @param locs the possible locations for the next move
   *  @return the location that was selected for the next move, or null to indicate
   *          that this OpossumCritter should be removed from the grid.
   */
  public Location selectMoveLocation(ArrayList<Location> locs)
  {  /* to be implemented in part (b) */  }
```

© 2008 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

```
  /** @param other the actor to check
   *  @return true if other is a friend; false otherwise
   */
  private boolean isFriend(Actor other)
  {  /* implementation not shown */  }

  /** @param other the actor to check
   *  @return true if other is a foe; false otherwise
   */
  private boolean isFoe(Actor other)
  {  /* implementation not shown */  }
}
```

© 2008 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

(a) Override the `processActors` method for the `OpossumCritter` class. This method should look at all elements of `actors` and determine whether or not to play dead according to the types of the actors. If there are more foes than friends, the `OpossumCritter` indicates that it is playing dead by changing its color to `Color.BLACK`. When not playing dead, it sets its color to `Color.ORANGE`. The instance variable `numStepsDead` should be updated to reflect the number of consecutive steps the `OpossumCritter` has played dead.

Complete method `processActors` below.

```
/** Whenever actors contains more foes than friends, this OpossumCritter plays dead.
 *  Postcondition: (1) The state of all actors in the grid other than this critter and the
 *  elements of actors is unchanged. (2) The location of this critter is unchanged.
 *  @param actors a group of actors to be processed
 */
public void processActors(ArrayList<Actor> actors)
```

(b) Override the `selectMoveLocation` method for the `OpossumCritter` class. When the `OpossumCritter` is not playing dead, it behaves like a `Critter`. The next location for an `OpossumCritter` that has been playing dead for three consecutive steps is `null`. Otherwise, an `OpossumCritter` that is playing dead remains in its current location.

Complete method `selectMoveLocation` below.

```
/** Selects the location for the next move.
 *  Postcondition: (1) The returned location is an element of locs, this critter's current location,
 *  or null. (2) The state of all actors is unchanged.
 *  @param locs the possible locations for the next move
 *  @return the location that was selected for the next move, or null to indicate
 *          that this OpossumCritter should be removed from the grid.
 */
public Location selectMoveLocation(ArrayList<Location> locs)
```

© 2008 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).