AP® Computer Science A
2005 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 4,700 schools, colleges, universities, and other educational organizations. Each year, the College Board serves over three and a half 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.

Copyright © 2005 by College Board. All rights reserved. College Board, AP Central, APCD, Advanced Placement Program, AP, AP Vertical Teams, Pre-AP, SAT, and the acorn logo are registered trademarks of the College Entrance Examination Board. Admitted Class Evaluation Service, CollegeEd, Connect to college success, MyRoad, SAT Professional Development, SAT Readiness Program, and Setting the Cornerstones are trademarks owned by the College Entrance Examination Board. PSAT/NMSQT is a registered trademark of the College Entrance Examination Board and National Merit Scholarship Corporation. Other products and services may be trademarks of their respective owners. Permission to use copyrighted College Board materials may be requested online at: http://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 and Pre-AP: 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.

Copyright © 2005 by College Entrance Examination Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for AP students and parents).

1. In this question, you will implement two methods for a class `Hotel` that is part of a hotel reservation system. The `Hotel` class uses the `Reservation` class shown below. A `Reservation` is for the person and room number specified when the `Reservation` is constructed.

```
public class Reservation
{
  public Reservation(String guestName, int roomNumber)
  { /* implementation not shown */ }

  public int getRoomNumber()
  { /* implementation not shown */ }

  // private data and other methods not shown
}
```

An incomplete declaration for the `Hotel` class is shown below. Each hotel in the hotel reservation system has rooms numbered 0, 1, 2, . . . , up to the last room number in the hotel. For example, a hotel with 100 rooms would have rooms numbered 0, 1, 2, . . . , 99.

```
public class Hotel
{
  private Reservation[] rooms;
    // each element corresponds to a room in the hotel;
    // if rooms[index] is null, the room is empty;
    // otherwise, it contains a reference to the Reservation
    // for that room, such that
    // rooms[index].getRoomNumber() returns index

  private ArrayList waitList;
    // contains names of guests who have not yet been
    // assigned a room because all rooms are full


  // if there are any empty rooms (rooms with no reservation),
  // then create a reservation for an empty room for the
  // specified guest and return the new Reservation;
  // otherwise, add the guest to the end of waitList
  // and return null
  public Reservation requestRoom(String guestName)
  { /* to be implemented in part (a) */ }


  // release the room associated with parameter res, effectively
  // canceling the reservation;
  // if any names are stored in waitList, remove the first name
  // and create a Reservation for this person in the room
  // reserved by res; return that new Reservation;
  // if waitList is empty, mark the room specified by res as empty and
  // return null
  // precondition:  res is a valid Reservation for some room
  //                in this hotel
  public Reservation cancelAndReassign(Reservation res)
  { /* to be implemented in part (b) */ }

  // constructors and other methods not shown
}
```

Copyright © 2005 by College Entrance Examination Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for AP students and parents).

(a) Write the `Hotel` method `requestRoom`. Method `requestRoom` attempts to reserve a room in the hotel for a given guest. If there are any empty rooms in the hotel, one of them will be assigned to the named guest and the newly created reservation is returned. If there are no empty rooms, the guest is added to the end of the waiting list and `null` is returned.

Complete method `requestRoom` below.

```
// if there are any empty rooms (rooms with no reservation),
// then create a reservation for an empty room for the
// specified guest and return the new Reservation;
// otherwise, add the guest to the end of waitList
// and return null
public Reservation requestRoom(String guestName)
```

(b) Write the `Hotel` method `cancelAndReassign`. Method `cancelAndReassign` releases a previous reservation. If the waiting list for the hotel contains any names, the vacated room is reassigned to the first person at the beginning of the list. That person is then removed from the waiting list and the newly created reservation is returned. If no one is waiting, the room is marked as empty and `null` is returned.

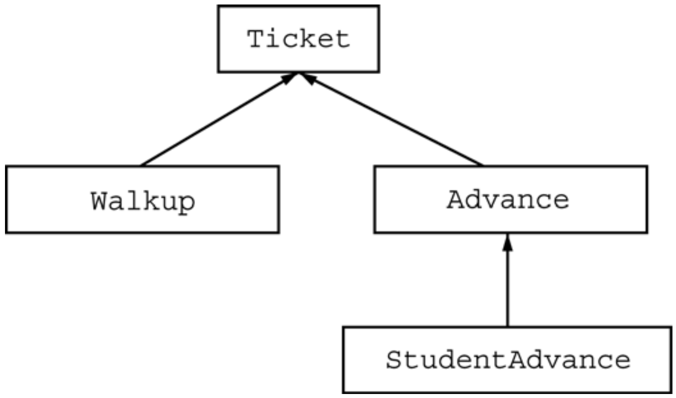
In writing `cancelAndReassign` you may call any accessible methods in the `Reservation` and `Hotel` classes. Assume that these methods work as specified.

Complete method `cancelAndReassign` below.

```
// release the room associated with parameter res, effectively
// canceling the reservation;
// if any names are stored in waitList, remove the first name
// and create a Reservation for this person in the room
// reserved by res; return that new Reservation;
// if waitList is empty, mark the room specified by res as empty and
// return null
// precondition:  res is a valid Reservation for some room
//                in this hotel
public Reservation cancelAndReassign(Reservation res)
```

Copyright © 2005 by College Entrance Examination Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for AP students and parents).

2. A set of classes is used to handle the different ticket types for a theater. The class hierarchy is shown in the following diagram.



All tickets have a serial number and a price. The class `Ticket` is specified as an `abstract` class as shown in the following declaration.

```
public abstract class Ticket
{
  private int serialNumber;   // unique ticket id number


  public Ticket()
  {  serialNumber = getNextSerialNumber();  }


  // returns the price for this ticket
  public abstract double getPrice();


  // returns a string with information about the ticket
  public String toString()
  {
    return "Number: " + serialNumber + "\nPrice: " + getPrice();
  }


  // returns a new unique serial number
  private static int getNextSerialNumber()
  {  /* implementation not shown */  }
}
```

Copyright © 2005 by College Entrance Examination Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for AP students and parents).

Each ticket has a unique serial number that is assigned when the ticket is constructed. For all ticket classes, the `toString` method returns a string containing the information for that ticket. Three additional classes are used to represent the different types of tickets and are described in the table below.

Class	Description	Sample `toString` Output
`Walkup`	These tickets are purchased on the day of the event and cost 50 dollars.	`Number: 712` `Price: 50`
`Advance`	Tickets purchased ten or more days in advance cost 30 dollars. Tickets purchased fewer than ten days in advance cost 40 dollars.	`Number: 357` `Price: 40`

`StudentAdvance`	These tickets are a type of `Advance` ticket that costs half of what that `Advance` ticket would normally cost.	`Number: 134`
		`Price: 15`
		`(student ID required)`

Using the class hierarchy and specifications given above, you will write complete class declarations for the `Advance` and `StudentAdvance` classes.

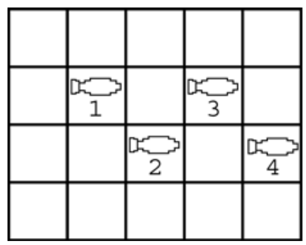
(a) Write the complete class declaration for the class `Advance`. Include all necessary instance variables and implementations of its constructor and method(s). The constructor should take a parameter that indicates the number of days in advance that this ticket is being purchased. Tickets purchased ten or more days in advance cost $30; tickets purchased nine or fewer days in advance cost $40.

(b) Write the complete class declaration for the class `StudentAdvance`. Include all necessary instance variables and implementations of its constructor and method(s). The constructor should take a parameter that indicates the number of days in advance that this ticket is being purchased. The `toString` method should include a notation that a student ID is required for this ticket. A `StudentAdvance` ticket costs half of what that `Advance` ticket would normally cost. If the pricing scheme for `Advance` tickets changes, the `StudentAdvance` price should continue to be computed correctly with no code modifications to the `StudentAdvance` class.

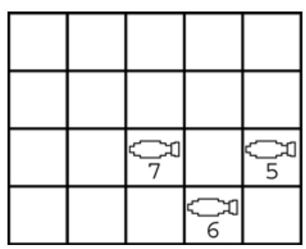
Copyright © 2005 by College Entrance Examination Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for AP students and parents).

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

Consider defining a new type of fish called `ZigZagFish`, which moves in a zigzag pattern. The first time a `ZigZagFish` moves, it will move to the location forward and to the right if that cell is empty. This is illustrated in the figure below as the move from position 1 to position 2. In each subsequent move, the `ZigZagFish` will attempt to move to the forward diagonal location on the side opposite its previous move (the second move will be to the left forward diagonal, the third move will be to the right forward diagonal, and so on). When the `ZigZagFish` has successfully moved, its direction does not change. If the `ZigZagFish` is unable to move, it stays in the same location but reverses its direction. After reversing its direction, the next time the `ZigZagFish` moves, it will attempt to move in the same diagonal direction as it tried before reversing. The diagrams below show the path followed by a single `ZigZagFish` object as a result of multiple moves.



Now consider what happens when the `ZigZagFish` attempts to move forward diagonally to the <u>left</u> from position 4. This move is blocked, and consequently the `ZigZagFish` stays in the same location but reverses its direction. This is illustrated as position 5 in the diagram below. From position 5, the `ZigZagFish` moves forward diagonally to the <u>left</u>, and from position 6, it moves forward diagonally to the <u>right</u> to position 7.



Copyright © 2005 by College Entrance Examination Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for AP students and parents).

GO ON TO THE NEXT PAGE.

The `ZigZagFish` class is defined by extending the `Fish` class and overriding the `move` and `nextLocation` methods. Because a `ZigZagFish` alternates its pattern of movement, a private instance variable `willZigRight` keeps track of the direction of the next movement.

The partial declaration for class `ZigZagFish` is shown below.

```
public class ZigZagFish extends Fish
{
  private boolean willZigRight;
    // true indicates the next move should be forward to the right
    // false indicates the next move should be forward to the left


  public ZigZagFish(Environment env, Location loc)
  {
    super(env, loc);
    willZigRight = true; // direction of the next move
  }


  // returns the forward diagonal cell to the left or right of this fish
  // (depending on willZigRight) if that cell is empty;
  // otherwise, returns this fish's current location
  // postcondition: the state of this ZigZagFish is unchanged
  protected Location nextLocation()
  {  /* to be implemented in part (a) */  }


  // moves this ZigZagFish diagonally (as specified in nextLocation) if
  // possible; otherwise, reverses direction without moving;
  // after a diagonal move, willZigRight is updated
  protected void move()
  {  /* to be implemented in part (b) */  }


  // other constructors, generateChild, and other methods not shown
}
```

Copyright © 2005 by College Entrance Examination Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for AP students and parents).

(a) Override the `nextLocation` method for the `ZigZagFish` class. The `nextLocation` method returns the cell diagonally forward to the right, the cell diagonally forward to the left, or the cell that the `ZigZagFish` currently occupies, according to the description given at the beginning of this question.

In writing `nextLocation`, you may use any of the accessible methods of the classes in the case study.

Complete method `nextLocation` below.

```
// returns the forward diagonal cell to the left or right of this fish
// (depending on willZigRight) if that cell is empty;
// otherwise, returns this fish's current location
// postcondition: the state of this ZigZagFish is unchanged
protected Location nextLocation()
```

(b) Override the `move` method for the `ZigZagFish` class. This method should change the location and direction of the fish as needed, according to the rules of movement described at the beginning of the question. In addition, the state of the fish must be updated.

In writing `move`, you may call `nextLocation`. Assume that `nextLocation` works as specified, regardless of what you wrote in part (a). You may also use any of the accessible methods of the classes in the case study.

Complete method `move` below.

```
// moves this ZigZagFish diagonally (as specified in nextLocation) if
// possible; otherwise, reverses direction without moving;
// after a diagonal move, willZigRight is updated
protected void move()
```

Copyright © 2005 by College Entrance Examination Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for AP students and parents).

4. Consider a grade-averaging scheme in which the final average of a student's scores is computed differently from the traditional average if the scores have "improved." Scores have improved if each score is greater than or equal to the previous score. The final average of the scores is computed as follows.

   A student has `n` scores indexed from 0 to `n-1`. If the scores have improved, only those scores with indexes greater than or equal to `n/2` are averaged. If the scores have not improved, all the scores are averaged.

   The following table shows several lists of scores and how they would be averaged using the scheme described above.

Student Scores	Improved?	Final Average
50, 50, 20, 80, 53	No	(50 + 50 + 20 + 80 + 53) / 5.0 = 50.6
20, 50, 50, 53, 80	Yes	(50 + 53 + 80) / 3.0 = 61.0
20, 50, 50, 80	Yes	(50 + 80) / 2.0 = 65.0

   Consider the following incomplete `StudentRecord` class declaration. Each `StudentRecord` object stores a list of that student's scores and contains methods to compute that student's final average.

```
public class StudentRecord
{
  private int[] scores; // contains scores.length values
                        // scores.length > 1

  // constructors and other data fields not shown


  // returns the average (arithmetic mean) of the values in scores
  // whose subscripts are between first and last, inclusive
  // precondition:  0 <= first <= last < scores.length
  private double average(int first, int last)
  {  /* to be implemented in part (a) */  }


  // returns true if each successive value in scores is greater
  // than or equal to the previous value;
  // otherwise, returns false
  private boolean hasImproved()
  {  /* to be implemented in part (b) */  }


  // if the values in scores have improved, returns the average
  // of the elements in scores with indexes greater than or equal
  // to scores.length/2;
  // otherwise, returns the average of all of the values in scores
  public double finalAverage()
  {  /* to be implemented in part (c) */  }

}
```

Copyright © 2005 by College Entrance Examination Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for AP students and parents).

(a) Write the `StudentRecord` method `average`. This method returns the average of the values in `scores` given a starting and an ending index.

Complete method `average` below.

```
// returns the average (arithmetic mean) of the values in scores
// whose subscripts are between first and last, inclusive
// precondition:  0 <= first <= last < scores.length
private double average(int first, int last)
```

(b) Write the `StudentRecord` method `hasImproved`.

Complete method `hasImproved` below.

```
// returns true if each successive value in scores is greater
// than or equal to the previous value;
// otherwise, returns false
private boolean hasImproved()
```

(c) Write the `StudentRecord` method `finalAverage`.

In writing `finalAverage`, you must call the methods defined in parts (a) and (b). Assume that these methods work as specified, regardless of what you wrote in parts (a) and (b).

Complete method `finalAverage` below.

```
// if the values in scores have improved, returns the average
// of the elements in scores with indexes greater than or equal
// to scores.length/2;
// otherwise, returns the average of all of the values in scores
public double finalAverage()
```

END OF EXAM

Copyright © 2005 by College Entrance Examination Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for AP students and parents).