AP® COMPUTER SCIENCE A
2006 SCORING GUIDELINES

Question 1: Daily Schedule

Part A: `conflictsWith` **1 1/2 points**

+1/2 call `OBJ1.overlapsWith(OBJ2)`
+1/2 access `getTime of other and this`
+1/2 return correct value

Part B: `clearConflicts` **3 points**

+2 loop over `apptList`
- **+1/2** reference `apptList` in loop body
- **+1/2** access appointment *in context of loop* (`apptList.get(i)`)
- **+1** access all appointments (cannot skip entries after a removal)

+1 remove conflicts *in context of loop*
- **+1/2** determine when conflict exists (must call `conflictsWith`)
- **+1/2** remove all conflicting appointments (and no others)

Part C: `addAppt` **4 1/2 points**

+1/2 test if emergency (*may limit to when emergency AND conflict exists*)
+1/2 clear conflicts if and only if emergency
(must not reimplement `clearConflicts` code)
+1/2 add `appt` if emergency

+2 non-emergency case
- **+1/2** loop over `apptList` (must reference `apptList` in body)
- **+1/2** access `apptList` element and check for `appt` conflicts *in context of loop*
- **+1/2** exit loop with state (conflict / no conflict) correctly determined *(includes loop bound)*
- **+1/2** add `appt` if and only if no conflict

+1 return true if any appointment added, false otherwise (must return both)

Usage: -1 if loop structure results in failure to handle empty `apptList`

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

AP® COMPUTER SCIENCE A/AB 2006 GENERAL USAGE

Most common usage errors are addressed specifically in rubrics with points deducted in a manner other than indicated on this sheet. The rubric takes precedence.

Usage points can only be deducted if the part where it occurs has earned credit.

A usage error that occurs once when the same usage is correct two or more times can be regarded as an oversight and not penalized. If the usage error is the only instance, one of two, or occurs two or more times, then it should be penalized.

A particular usage error should be penalized only once in a problem, even if it occurs on different parts of a problem.

Nonpenalized Errors

spelling/case discrepancies*

local variable not declared when any other variables are declared in some part

default constructor called without parens; for example, `new Fish;`

use keyword as identifier

`[r,c]`, `(r)(c)` or `(r,c)` instead of `[r][c]`

= instead of == (and vice versa)

`length`/`size` confusion for array, `String`, and `ArrayList`, with or without `()`

`private` qualifier on local variable

extraneous code with no side-effect, for example a check for precondition

common mathematical symbols for operators (x • ÷ ≤ ≥ <> ≠)

missing { } where indentation clearly conveys intent

missing `( )` on method call or around `if/while` conditions

missing `;`s

missing “new” for constructor call once, when others are present in some part

missing downcast from collection

missing `int` cast when needed

missing `public` on class or constructor header

Minor Errors (1/2 point)

confused identifier (e.g., `len` for `length` or `left()` for `getLeft()`)

no local variables declared

`new` never used for constructor calls

`void` method or constructor returns a value

modifying a constant (`final`)

use `equals` or `compareTo` method on primitives, for example `int x; …x.equals(val)`

`[] – get` confusion if access not tested in rubric

assignment dyslexia, for example, `x + 3 = y;` for `y = x + 3;`

`super(method())` instead of `super.method()`

formal parameter syntax (with type) in method call, e.g., `a = method(int x)`

missing `public` from method header when required

"false"/"true" or 0/1 for boolean values

"null" for `null`

Major Errors (1 point)

extraneous code which causes side-effect, for example, information written to output

use interface or class name instead of variable identifier, for example `Simulation.step()` instead of `sim.step()`

`aMethod(obj)` instead of `obj.aMethod()`

use of object reference that is incorrect, for example, use of `f.move()` inside method of `Fish` class

use private data or method when not accessible

destruction of data structure (e.g., by using root reference to a `TreeNode` for traversal of the tree)

use class name in place of `super` either in constructor or in method call

**Note: Spelling and case discrepancies for identifiers fall under the "nonpenalized" category as long as the correction can be unambiguously inferred from context. For example, "Queu" instead of "Queue". Likewise, if a student declares "Fish fish;", then uses Fish.move() instead of fish.move(), the context allows for the reader to assume the object instead of the class.*

AP® COMPUTER SCIENCE A
2006 CANONICAL SOLUTIONS

Question 1: Daily Schedule

PART A:

```
public boolean conflictsWith(Appointment other)
{
  return getTime().overlapsWith(other.getTime());
}
```

PART B:

```
public void clearConflicts(Appointment appt)
{
  int i = 0;
  while (i < apptList.size())
  {
    if (appt.conflictsWith((Appointment)(apptList.get(i))))
    {
      apptList.remove(i);
    }
    else
    {
      i++;
    }
  }
}
```

ALTERNATE SOLUTION

```
public void clearConflicts(Appointment appt)
{
  for (int i = apptList.size()-1; i >= 0; i--)
  {
    if (appt.conflictsWith((Appointment)apptList.get(i)))
    {
      apptList.remove(i);
    }
  }
}
```

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

Question 1: Daily Schedule (continued)

PART C:

```
public boolean addAppt(Appointment appt, boolean emergency)
{
  if (emergency)
  {
    clearConflicts(appt);
  }
  else
  {
    for (int i = 0; i < apptList.size(); i++)
    {
      if (appt.conflictsWith((Appointment)apptList.get(i)))
      {

        return false;
      }
    }
  }
  return apptList.add(appt);
```

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

Al A,

(a) Write the `Appointment` method `conflictsWith`. If the time interval of the current appointment overlaps with the time interval of the appointment `other`, method `conflictsWith` should return `true`, otherwise, it should return `false`.

Complete method `conflictsWith` below.

```
// returns true if the time interval of this Appointment
// overlaps with the time interval of other;
// otherwise, returns false
public boolean conflictsWith(Appointment other)
{  TimeInterval thisApp = getTime(); /* There should be a field but it doesn't show*/
   TimeInterval otherApp = other.getTime;
   return thisApp.overlapsWith(otherApp);
}
```

Part (b) begins on page 6.

GO ON TO THE NEXT PAGE.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

AlA 2

Complete method `clearConflicts` below.

```
// removes all appointments that overlap the given Appointment
// postcondition: all appointments that have a time conflict with
//                appt have been removed from this DailySchedule
public void clearConflicts(Appointment appt)
{   for(int i=0; i<apptList.size(); i++)
  {Appointment oldApp = (Appointment)apptList.get(i);
      if (appt.conflictsWith(oldApp))
      {  apptList.remove(i);
          i--;
      }
  }
}
```

Part (c) begins on page 8.

GO ON TO THE NEXT PAGE.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

(c) Write the `DailySchedule` method `addAppt`. The parameters to method `addAppt` are an appointment and a `boolean` value that indicates whether the appointment to be added is an emergency. If the appointment is an emergency, the schedule is cleared of all appointments that have a time conflict with the given appointment and the appointment is added to the schedule. If the appointment is not an emergency, the schedule is checked for any conflicting appointments. If there are no conflicting appointments, the given appointment is added to the schedule. Method `addAppt` returns `true` if the appointment was added to the schedule; otherwise, it returns `false`.

In writing method `addAppt`, you may assume that `conflictsWith` and `clearConflicts` work as specified, regardless of what you wrote in parts (a) and (b).

Complete method `addAppt` below.

```
// if emergency is true, clears any overlapping appointments and adds
// appt to this DailySchedule; otherwise, if there are no conflicting
// appointments, adds appt to this DailySchedule;
// returns true if the appointment was added;
// otherwise, returns false
public boolean addAppt(Appointment appt, boolean emergency)
{ if (emergency)
   { clearConflicts(appt);
     apptList.add(appt);
     return true;
   }
   else
   {  int check = 0;
      for (int i=0; i< apptList.size(); i++)
      { if( appt. conflictsWith((Appointment)apptList.get(i))
         { check++;}
      }
      if (check >0)
      { return false;}
      else
      { apptList.add(appt);
         return true;
      }
```

GO ON TO THE NEXT PAGE.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

A1B,

(a) Write the `Appointment` method `conflictsWith`. If the time interval of the current appointment overlaps with the time interval of the appointment `other`, method `conflictsWith` should return `true`, otherwise, it should return `false`.

Complete method `conflictsWith` below.

```
// returns true if the time interval of this Appointment
// overlaps with the time interval of other;
// otherwise, returns false
public boolean conflictsWith(Appointment other){
    if(getTime().compareTo(other.getTime)==0){
        return true;
    }
    else{
        return false;
    }

}
```

Part (b) begins on page 6.

GO ON TO THE NEXT PAGE.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

A1 B2

Complete method `clearConflicts` below.

```
  // removes all appointments that overlap the given Appointment
  // postcondition: all appointments that have a time conflict with
  //                appt have been removed from this DailySchedule
  public void clearConflicts(Appointment appt){
      for(int i=0; i<apptList.size(); i++){
          if(apptList.get(i).conflictsWith(appt)){
              apptList.remove(i);
          }
      }
  }
```

Part (c) begins on page 8.

GO ON TO THE NEXT PAGE.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

(c) Write the `DailySchedule` method `addAppt`. The parameters to method `addAppt` are an appointment and a `boolean` value that indicates whether the appointment to be added is an emergency. If the appointment is an emergency, the schedule is cleared of all appointments that have a time conflict with the given appointment and the appointment is added to the schedule. If the appointment is not an emergency, the schedule is checked for any conflicting appointments. If there are no conflicting appointments, the given appointment is added to the schedule. Method `addAppt` returns `true` if the appointment was added to the schedule; otherwise, it returns `false`.

In writing method `addAppt`, you may assume that `conflictsWith` and `clearConflicts` work as specified, regardless of what you wrote in parts (a) and (b).

Complete method `addAppt` below.

```
// if emergency is true, clears any overlapping appointments and adds
// appt to this DailySchedule; otherwise, if there are no conflicting
// appointments, adds appt to this DailySchedule;
// returns true if the appointment was added;
// otherwise, returns false
public boolean addAppt(Appointment appt, boolean emergency){
    if(emergency){
        clearConflicts(appt);
        apptList.add(appt);
        return true;
    }
    else{
        return false;
    }
```

GO ON TO THE NEXT PAGE.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

A1 C1

(a) Write the Appointment method conflictsWith. If the time interval of the current appointment overlaps with the time interval of the appointment other, method conflictsWith should return true, otherwise, it should return false.

Complete method conflictsWith below.

```
// returns true if the time interval of this Appointment
// overlaps with the time interval of other;
// otherwise, returns false
public boolean conflictsWith(Appointment other) {
  TimeInterval ti = new TimeInterval();
    return (ti.overlapsWith(other.getTime()));

}
```

Part (b) begins on page 6.

GO ON TO THE NEXT PAGE.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

AI C2

Complete method clearConflicts below.

```
// removes all appointments that overlap the given Appointment
// postcondition: all appointments that have a time conflict with
//                appt have been removed from this DailySchedule
public void clearConflicts(Appointment appt)
    TimeInteroal q = new TimeInteroal();
    for(int check = 0; check < apptList.size; check++)
    {
        if(q. ConflictsWith(appt)
        {
            applist. remove(check);
        }
    }
    }
```

Part (c) begins on page 8.

GO ON TO THE NEXT PAGE.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

A1 C3

(c) Write the `DailySchedule` method `addAppt`. The parameters to method `addAppt` are an appointment and a `boolean` value that indicates whether the appointment to be added is an emergency. If the appointment is an emergency, the schedule is cleared of all appointments that have a time conflict with the given appointment and the appointment is added to the schedule. If the appointment is not an emergency, the schedule is checked for any conflicting appointments. If there are no conflicting appointments, the given appointment is added to the schedule. Method `addAppt` returns `true` if the appointment was added to the schedule; otherwise, it returns `false`.

In writing method `addAppt`, you may assume that `conflictsWith` and `clearConflicts` work as specified, regardless of what you wrote in parts (a) and (b).

Complete method `addAppt` below.

```
// if emergency is true, clears any overlapping appointments and adds
// appt to this DailySchedule; otherwise, if there are no conflicting
// appointments, adds appt to this DailySchedule;
// returns true if the appointment was added;
// otherwise, returns false
public boolean addAppt(Appointment appt, boolean emergency) {
    Time Interval Time = new timeInterval();
    if (emergency)
    {
  if (Time. overlapsWith(appt.getTime() );
        {
            clearConflicts(appt);
            apptList.add(appt);
        }

    return false;
    }

}
```

GO ON TO THE NEXT PAGE.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

Question 1

Overview

This question focused on abstraction and data structure access. It involved storing and manipulating appointments, each having a time interval associated with it. In part (a) students were required to complete the `conflictsWith` method in the provided `Appointment` class, so that it compared the current appointment with another appointment and determined whether they overlapped. This involved accessing the underlying time interval for the two appointments and calling the appropriate method from the `TimeInterval` class to see if an overlap occurred. A `DailySchedule` class was then provided that stored an `ArrayList` of `Appointment` objects in a private data field. In part (b) students were required to complete the `clearConflicts` method of this class, which involved traversing the `ArrayList`, identifying any appointments that conflicted with the specified appointment (by calling the `conflictsWith` method from part (a)), and removing all conflicting appointments. In part (c) students were required to complete the `addAppt` method, which attempted to add a new appointment to the daily schedule. This involved traversing the `ArrayList` to determine if any conflicts occurred, removing conflicts in the case of an emergency priority, and adding the new appointment as long as no conflicts remained.

Sample: A1A
Score: 9

This solution earned full credit for all three parts. Its `conflictsWith` method correctly uses the methods `getTime` and `overlapsWith` to return `true` if and only if the time interval of the current appointment overlaps with the time interval of the other appointment. Its `clearConflicts` method examines the element that immediately follows a removed one by decrementing the loop counter whenever an element is removed. Its `addAppt` method tests for an emergency and in the emergency case clears the conflicts, adds the appointment, and returns `true`. In the non-emergency case, it initializes the check counter to 0 and then increments it by one every time a conflict is found. If after comparing the appointment with each appointment in the list, any conflicts have been found (`check` is positive), `false` is returned; otherwise the appointment is added and `true` is returned.

Sample: A1B
Score: 5

This solution earned a ½ point for its `conflictsWith` method, which gets the time of both the current appointment and the appointment `other`. The method uses a `compareTo` method instead of `overlapsWith`. The solution earned 2 points for its `clearConflicts` method, which is correct except that it does not examine the element that immediately follows a removed one. The solution earned two ½ points for its `addAppt` method. The method tests for an emergency and in the emergency case clears the conflicts, adds the appointment, and returns `true`. In the non-emergency case it simply returns `false`, neither checking for conflicts, nor attempting to add an appointment.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).

Question 1 (continued)

Sample: A1C
Score: 2

This solution earned a ½ point for its `conflictsWith` method, which applies `overlapsWith` to two objects but does not access the time interval of the current appointment. The solution earned 1 point for its `clearConflicts` method, which attempts to loop over the `ArrayList` instance field `apptList`, but does not access any elements. The method applies `conflictsWith` to a new time interval instead of an appointment from the list. It correctly uses the `ArrayList` method `remove`. The solution earned a ½ point for its `addAppt` method, which tests for an emergency and does nothing in the non-emergency case. In the emergency case, the statements are guarded by a test with an indeterminable value.

© 2006 The College Board. All rights reserved.
Visit apcentral.collegeboard.com (for AP professionals) and www.collegeboard.com/apstudents (for students and parents).