AP® Computer Science A
2007 Scoring Guidelines

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.

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

AP® COMPUTER SCIENCE A
2007 SCORING GUIDELINES

Question 1: Self Divisor

Part A: isSelfDivisor **4 points**

+2 loop over digits
- **+1** access digit *in context of loop*
  - **+1/2** attempt (`number % ?` or successfully convert to string represent)
  - **+1/2** correct
- **+1** process all digits
  - **+1/2** attempt (process multiple digits)
  - **+1/2** correct

+2 classify `number`
- **+1/2** return false if find 0 digit
- **+1/2** test if divisible (`number %` digit)
- **+1/2** return false if find non-divisor digit
- **+1/2** return true self divisor

lose both of these if return a value in both cases of an if-else

Part B: firstNumSelfDivisors **5 points**

+1 initialize
- **+1/2** create and initialize array of size `num`
- **+1/2** create and initialize index counter

+3 1/2 loop to find self divisors
- **+1/2** iterate through numbers beginning with `start`
- **+1/2** call `isSelfDivisor` on number
- **+1 1/2** add self divisor to array
  - **+1/2** attempt (store self divisor in some array index)
  - **+1** correct (store in correct index, including increment)
- **+1** loop and store `num` values in array
  - **+1/2** attempt (must reference index counter and `num`)
  - **+1/2** correct

+1/2 return array (lose this if return first time through loop)

AP® COMPUTER SCIENCE A
2007 SCORING GUIDELINES

Question 2: Pounce Fish (MBS)

Part A: findFish **5 points**

+2 access & check neighbor
- **+1/2** determine current location
- **+1/2** determine current direction
- **+1/2** correctly access any neighbor
- **+1/2** determine if neighbor location is empty

+1 1/2 loop in forward direction
- **+1/2** loop with respect to range
- **+1** access up to `range` consecutive forward locations (as needed)

+ 1 1/2 return value
- **+1/2** return null if reach invalid location in loop
- **+1/2** return object at first non-empty location in loop
- **+1/2** return null if no non-empty location in loop

Special Usage:

-1 missing or incorrect environment access

Part B: act **4 points**

+1/2 call `findFish()`

+1/2 test if `findFish` returned null

+2 not null case
- **+1** `prey.die()` or `environment().remove(prey)`
- **+1** change location to prey's location

+1 null case
- **+1/2** attempt to act (`move()` or `super.move()` OK)
- **+1/2** `super.act()`

AP® COMPUTER SCIENCE A
2007 SCORING GUIDELINES

Question 3: Answer Sheets

Part A: getScore **4 1/2 points**

+1/2 initialize score (a double) or right/wrong counters

+1 1/2 loop over either `answers` or `key`
- **+1/2** reference `answers` or `key` in loop body
- **+1/2** correctly access `answers` or `key` element in loop body
- **+1/2** access all `answers` or `key` elements

+2 calculate score
- **+1/2** attempt to compare an `answers` element and a `key` element (== ok)
- **+1/2** correctly compare corresponding elements using `equals`
- **+1/2** add 1 to score if and only if equal
- **+1/2** subtract 1/4 from score if and only if not equal and answer not "?"

+1/2 return calculated score

Part B: highestScoringStudent **4 1/2 points**

+1 1/2 loop over `sheets`
- **+1/2** reference `sheets` in loop body
- **+1/2** correctly access `sheets` element in context of loop
- **+1/2** access all elements of `sheets`

+2 determine highest score
- **+1/2** get student score (call `getScore(key)` on a `sheets` element)
- **+1/2** compare student score with highest so far (in context of loop)
- **+1** correctly identify highest score (lose this if use constant for initial high)

+1 return name
- **+1/2** access name (call `getName` on highest)
- **+1/2** return name

AP® COMPUTER SCIENCE A
2007 SCORING GUIDELINES

Question 4: Game Design (Design)

Part A: RandomPlayer **4 points**

+1/2 `class RandomPlayer extends Player`

+1 constructor
- **+1/2** `public RandomPlayer(String aName)`
- **+1/2** `super(aName)`

+2 1/2 `getNextMove`
- **+1/2** `state.getCurrentMoves()`
- **+1** if no moves
  - **+1/2** test if size = 0
  - **+1/2** return "no move" only if 0 moves
- **+1** if moves
  - **+1/2** select random move index
  - **+1/2** return random move

Part B: play **5 points**

+1/2 print initial state (OK to print in loop)

+3 make repeated moves
- **+1** repeat until `state.isGameOver()`
- **+1/2** `state.getCurrentPlayer()`
- **+1/2** `player.getNextMove(state)`
- **+1/2** display player and move
- **+1/2** make move

+1 1/2 determine winner
- **+1/2** `state.getWinner()`
- **+1/2** display message if draw (if `getWinner` returns null)
- **+1/2** display message if winner

lose both if done before game ends

AP® COMPUTER SCIENCE A
2007 CANONICAL SOLUTIONS

Question 1: Self Divisor

PART A:

```
public static boolean isSelfDivisor(int number) {
     int n = number;
     while (n > 0) {
         int digit = n % 10;
         if (digit == 0 || number % digit != 0) {
             return false;
         }
         n /= 10;
     }
     return true;
}
```

ALTERNATE SOLUTION:

```
public static boolean isSelfDivisor(int number) {
     String str = "" + number;
     for (int i = 0; i < str.length(); i++) {
         int digit = Integer.parseInt(str.substring(i,i+1));
         if (digit == 0 || number % digit != 0) {
             return false;
         }
     }
     return true;
}
```

PART B:

```
public static int[] firstNumSelfDivisors(int start, int num) {
    int[] selfs = new int[num];
    int numStored = 0;
    int nextNumber = start;
    while (numStored < num) {
        if (isSelfDivisor(nextNumber)) {
            selfs[numStored] = nextNumber;
            numStored++;
        }
        nextNumber++;
    }
    return selfs;
}
```

ALTERNATE SOLUTION:

```
public static int[] firstNumSelfDivisors(int start, int num) {
    int[] selfs = new int[num];
    int numStored = 0;
    int nextNumber = start;
    for (int i = 0; i < num; i++) {
        while (!isSelfDivisor(nextNumber)) {
            nextNumber++;
        }
        selfs[numStored] = nextNumber;
        numStored++;
        nextNumber++;
    }
    return selfs;
}
```

AP® COMPUTER SCIENCE A
2007 CANONICAL SOLUTIONS

Question 2: Pounce Fish (MBS)

PART A:

```
private Fish findFish()
{
    Environment env = environment();
    Location loc = location();
    Direction dir = direction();

    for (int i = 0; i < range; i++) {
        loc = env.getNeighbor(loc, dir);
        if (!env.isEmpty(loc)) {
            return (Fish)env.objectAt(loc);
        }
    }
    return null;
}
```

PART B:

```
public void act()
{
    if (! isInEnv() )
        return;

    Fish prey = findFish();
    if (prey != null) {
        prey.die();             // OR environment().remove(prey);
        changeLocation(prey.location());
    }
    else {
        super.act();
    }
}
```

AP® COMPUTER SCIENCE A
2007 CANONICAL SOLUTIONS

Question 3: Answer Sheets

PART A:

```
public double getScore(ArrayList<String> key)
{
    double score = 0.0;
    for (int i = 0; i < answers.size(); i++) {
        if (answers.get(i).equals(key.get(i))) {
            score += 1.0;
        }
        else if (!answers.get(i).equals("?")) {
            score -= 0.25;
        }
    }
    return score;
}
```

PART B:

```
public String highestScoringStudent(ArrayList<String> key)
{
    StudentAnswerSheet highest = sheets.get(0);
    for (StudentAnswerSheet sheet : sheets) {
        if (sheet.getScore(key) > highest.getScore(key)) {
            highest = sheet;
        }
    }
    return highest.getName();
}
```

AP® COMPUTER SCIENCE A
2007 CANONICAL SOLUTIONS

Question 4: Game Design (Design)

PART A:

```
public class RandomPlayer extends Player
{
    public RandomPlayer(String aName)
    {
        super(aName);
    }

    public String getNextMove(GameState state)
    {
        ArrayList<String> possibleMoves = state.getCurrentMoves();
        if (possibleMoves.size() == 0) {
            return "no move";
        }
        else {
            int randomIndex = (int)(Math.random()*possibleMoves.size());
            return possibleMoves.get(randomIndex);
        }
    }
}
```

PART B:

```
public void play()
{
    System.out.println("Initial state:" + state);

    while (!state.isGameOver()) {
        Player currPlayer = state.getCurrentPlayer();
        String currMove = currPlayer.getNextMove(state);
        System.out.println(currPlayer.getName() + ": " + currMove);
        state.makeMove(currMove);
    }

    Player winner = state.getWinner();
    if (winner != null) {
        System.out.println(winner.getName() + " wins");
    }
    else {
        System.out.println("Game ends in a draw");
    }
}
```