AP® Computer Science AB
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 AB
2007 SCORING GUIDELINES

Question 1: Sliding Puzzle (Design)

Part A:	isDone	**3 1/2 points**

+1 1/2 2-D traversal
- **+1/2** correctly access an element of `values` in context of loop
- **+1** access all side^2 values (lose this if index out-of-bounds)

+2 classify (must check `values`)
- **+1** return false if numbers out of order
- **+1** return true if numbers in order (ignoring 0)

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

+1 1/2 pick random element from `temp`
- **+1** generate random int in range 0..`temp.size()`
- **+1/2** access element at the generated index

+1 1/2 store element in `values`
- **+1** assign accessed element to any index
- **+1/2** correctly assign to empty index

+1/2 `temp.remove(index)`

+1 assign & remove all side^2 values

Part C:	$O(n^2)$	**1/2 point**

Part D:	O(n)	**1/2 point**

AP® COMPUTER SCIENCE AB
2007 SCORING GUIDELINES

Question 2: Pair Matcher

Part A: constructor **5 points**

+1/2 `personMap = new HashMap<Person, PriorityQueue<Pair>>();`

+1 iterate over `personList`
- **+1/2** access a `personList` element in loop body
- **+1/2** access all `personList` elements

+2 construct priority queue
- **+1/2** `new PriorityQueue<Pair>()`
- **+1 1/2** add pairs to priority queue
  - **+1/2** nested iteration through entire `PersonList`
  - **+1/2** construct `Pair` object containing 2 `Person`s
  - **+1/2** add pair to the priority queue (but not duplicate)

+1 1/2 Put priority queues into map
- **+1/2** put at least one priority queue into `personMap`
- **+1** put a unique priority queue for every person in `PersonList`

Part B: removeNumMatches **4 points**

+1/2 `personMap.get(p)`

+1/2 return null if `p` is not in `personMap`

+2 1/2 store persons in array
- **+1/2** `new Person[num]`
- **+1** remove and access
  - **+1/2** remove pair from front of queue
  - **+1/2** add `person2` from pair to array
- **+1** repeat to add exactly `num` persons to array

+1/2 return array of persons

AP® COMPUTER SCIENCE AB
2007 SCORING GUIDELINES

Question 3: Tree Ball

Part A: getMaxHelper **4 points**

+1 base case
- **+1/2** test if `current == null`
- **+1/2** return 0

+3 recursive case
- **+1/2** `getMaxHelper` of left subtree
- **+1/2** `getMaxHelper` of right subtree
- **+1 1/2** calculate max
  - **+1/2** `current.getValue()`
  - **+1/2** determine max of left and right subtree max path scores
  - **+1/2** add root value to larger subtree max
- **+1/2** return sum of root value and max score of subtrees

Part B: constructor **5 points**

+2 create & init node
- **+1/2** create new `TreeNode`
- **+1/2** generate random digit 0...9
- **+1/2** store generated value in node
- **+1/2** assign references to left & right

+2 construct tree
- **+1** construct a multi-level tree with both left and right children
- **+1** construct full tree with `numLevel` levels

+1 assign constructed tree to `root`

AP® COMPUTER SCIENCE AB
2007 SCORING GUIDELINES

Question 4: Environment Iterator (MBS)

Part A: next **5 points**

+1/2 save current value of `loc`

+1/2 correctly access `loc.row()` and `loc.col()`

+1 bottom edge case
- **+1/2** determine if last row
- **+1/2** `new Location(loc.col()+1, env.numCols()-1);`

+1 left edge & non-bottom edge case
- **+1/2** determine if leftmost column (not on last row)
- **+1/2** `new Location(0, loc.row()+1);`

+1/2 otherwise, `new Location(loc.row()+1, loc.col()-1);`

+1 correctly assign `loc` in all three cases

+1/2 return saved `loc`

Part B: emptyLocs **4 points**

+1/2 create list of locations

+1/2 `EnvIterator iter = new EnvIterator(env);`

+2 1/2 add empty locations to list
- **+1/2** stop adding if `!iter.hasNext()`
- **+1/2** stop adding if `n` distinct locations added
- **+1/2** `iter.next()` in context of loop
- **+1** add empty location
  - **+1/2** check if location from iterator is empty
  - **+1/2** append empty location to list

+1/2 return the list of empty locations

AP® Computer Science AB
2007 Canonical Solutions

Question 1: Sliding Puzzle (Design)

PART A:

```
public boolean isDone()
{
    int nextAns = 1;
    for (int[] nextRow : values) {
        for (int nextVal : nextRow) {
            if (nextVal == nextAns) {
                nextAns++;
            }
            else if (nextVal != 0) {
                return false;
            }
        }
    }
    return true;
}
```

PART B:

```
public void initialize()
{
    ArrayList<Integer> temp = new ArrayList<Integer>();
    for (int j = 0; j < side*side; j++)
        temp.add(new Integer(j));

    for (int r = 0; r < side; r++) {
        for (int c = 0; c < side; c++) {
            int randIndex = (int)(Math.random()*temp.size());
            values[r][c] = temp.get(randIndex);
            temp.remove(randIndex);
        }
    }
}
```

PART C:

$O(n^2)$

PART D:

$O(n)$

AP® Computer Science AB
2007 Canonical Solutions

Question 2: Pair Matcher

PART A:

```
public PairMatcher(List<Person> personList)
{
    personMap = new HashMap<Person, PriorityQueue<Pair>>();
    for (Person p : personList) {
        PriorityQueue<Pair> queue = new PriorityQueue<Pair>();
        for (Person p1 : personList) {
            if (p != p1) {
                queue.add(new Pair(p, p1));
            }
        }
        personMap.put(p, queue);
    }
}
```

PART B:

```
public Person[] removeNumMatches(Person p, int num)
{
    PriorityQueue<Pair> queue = personMap.get(p);
    if (queue == null) {
        return null;
    }

    Person[] matches = new Person[num];
    for (int i = 0; i < num; i++) {
        matches[i] = queue.remove().getPerson2();
    }
    return matches;
}
```

AP® Computer Science AB
2007 Canonical Solutions

Question 3: Tree Ball

PART A:

```
private int getMaxHelper(TreeNode current)
{
    if (current == null) {
        return 0;
    }
    else {
        int leftMax = getMaxHelper(current.getLeft());
        int rightMax = getMaxHelper(current.getRight());
        if (leftMax >= rightMax) {
            return ((Integer)current.getValue()).intValue() + leftMax;
        }
        else {
            return ((Integer)current.getValue()).intValue() + rightMax;
        }
    }
}
```

PART B:

```
public GameBoard(int numLevels)
{
    root = GameBoardHelper(numLevels);
}

private TreeNode GameBoardHelper(int numLevels)
{
    if  (numLevels == 0) {
        return null;
    }
    else {
        return new TreeNode(new Integer((int)(Math.random()*10)),
                            GameBoardHelper(numLevels-1),
                            GameBoardHelper(numLevels-1));
    }
}
```

AP® Computer Science AB
2007 Canonical Solutions

Question 4: Environment Iterator (MBS)

PART A:

```
public Location next()
{
    Location retLoc = loc;
    if (loc.row() == env.numRows()-1) {
        loc = new Location(loc.col()+1 , env.numCols()-1);
    }
    else if (loc.col() == 0) {
        loc = new Location(0, loc.row()+1);
    }
    else {
        loc = new Location(loc.row()+1, loc.col()-1);
    }
    return retLoc;
}
```

PART B:

```
public List<Location> emptyLocs(BoundedEnv env, int n)
{
    List<Location> empties = new ArrayList<Location>();

    EnvIterator iter = new EnvIterator(env);
    while (iter.hasNext() && empties.size() < n) {
        Location next = iter.next();
        if (env.isEmpty(next)) {
            empties.add(next);
        }
    }
    return empties;
}
```