AP® Computer Science A
2009 Canonical Solutions

The College Board

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 5,600 schools, colleges, universities and other educational organizations. Each year, the College Board serves seven million students and their parents, 23,000 high schools and 3,800 colleges through major programs and services in college readiness, 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 AP teachers: apcentral.collegeboard.com.

AP® COMPUTER SCIENCE A
2009 CANONICAL SOLUTIONS

Question 1: Number Cube

PART A:

```
/** Returns an array of the values obtained by tossing
 *    a number cube numTosses times.
 *  @param cube a NumberCube
 *  @param numTosses the number of tosses to be recorded
 *         Precondition: numTosses > 0
 *  @return an array of numTosses values
 */
public static int[] getCubeTosses(NumberCube cube, int numTosses)
{
  int[] cubeTosses = new int[numTosses];
  for (int i = 0; i < numTosses; i++)
  {
    cubeTosses[i] = cube.toss();
  }
  return cubeTosses;
}
```

These canonical solutions serve an expository role, depicting general approaches to a solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.

AP® COMPUTER SCIENCE A
2009 CANONICAL SOLUTIONS

Question 1: Number Cube (continued)

PART B:

```
/** Returns the starting index of a longest run of two or more
 *    consecutive repeated values in the array values.
 *  @param values an array of integer values representing a series
 *    of number cube tosses
 *    Precondition: values.length > 0
 *  @return the starting index of a run of maximum size;
 *          -1 if there is no run
 */
public static int getLongestRun(int[] values)
{
  int currentLen = 0;
  int maxLen = 0;
  int maxStart = -1;
  for (int i = 0; i < values.length-1; i++)
  {
    if (values[i] == values[i+1])
    {
      currentLen++;
      if (currentLen > maxLen)
      {
        maxLen = currentLen;
        maxStart = i - currentLen + 1;
      }
    }
    else
    {
      currentLen = 0;
    }
  }
  return maxStart;
}
```

These canonical solutions serve an expository role, depicting general approaches to a solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.

AP® COMPUTER SCIENCE A
2009 CANONICAL SOLUTIONS

Question 1: Number Cube (continued)

PART B (ALTERNATE SOLUTION I):

```
public static int getLongestRun(int[] values)
{
  int maxStart = -1;
  int maxLen = -1;
  int currentLen = 0;
  int currVal = -1;
  int currStart = 0;
  for (int i = 0; i < values.length; i++)
  {
    if (values[i] == currVal)
      currentLen++;
    else
    {
      if (currentLen > maxLen)
      {
        maxLen = currentLen;
        maxStart = currStart;
      }
      currStart = i;
      currentLen = 1;
      currVal=values[i];
    }
  }

  if (currentLen > maxLen)
  {
    maxLen = currentLen;
    maxStart = currStart;
  }
  if (maxLen == 1)
    return -1;
  else
    return maxStart;
}
```

These canonical solutions serve an expository role, depicting general approaches to a solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.

AP® COMPUTER SCIENCE A
2009 CANONICAL SOLUTIONS

Question 1: Number Cube (continued)

PART B (ALTERNATE SOLUTION II):

```
public static int getLongestRun(int[] values)
{
  int maxLen = 0;
  int currLen = 0;
  int index = -1;
  int currVal = -1;
  for (int i = values.length - 1; i >= 0; i--)
  {
    if (values[i] == currVal)
      currLen++;
    else
    {
      if (maxLen < currLen)
      {
        maxLen = currLen;
        index = i+1;
      }
      currVal = values[i];
      currLen = 1;
    }
  }

  if (maxLen < currLen)
  {
    maxLen = currLen;
    index = 0;
  }
  if (maxLen == 1)
    return -1;

  return index;
}
```

These canonical solutions serve an expository role, depicting general approaches to a solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.

AP® COMPUTER SCIENCE A
2009 CANONICAL SOLUTIONS

Question 2: Stockpile Critter

```
public class StockpileCritter extends Critter
{
  /** Energy stockpile, initialized to 0. */
  //  Instance variable auto-initialized so =0 not necessary.
  private int stockpile = 0;

  /** Default constructor sufficient; no constructors needed. */
  // public StockpileCritter() {stockpile = 0;}


  /** Overridden to address stockpile behavior. */
  public void processActors(ArrayList<Actor> actors)
  {
    this.stockpile += actors.size();
    for (Actor a : actors)
      a.removeSelfFromGrid();

    this.stockpile--;
  }


  /** Overridden to address stockpile behavior. */
  public Location selectMoveLocation(ArrayList<Location> locs)
  {
    if (this.stockpile < 0)
      return null;
    else
      return super.selectMoveLocation(locs);
  }
}
```

These canonical solutions serve an expository role, depicting general approaches to a solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.

AP® COMPUTER SCIENCE A
2009 CANONICAL SOLUTIONS

Question 3: Battery Charger

PART A:

```
/** Determines the total cost to charge the battery starting
 *    at the beginning of startHour.
 *  @param startHour the hour at which the charge period begins
 *         Precondition: 0 ≤ startHour ≤ 23
 *  @param chargeTime the number of hours the battery needs to be charged
 *         Precondition: chargeTime > 0
 *  @return the total cost to charge the battery
 */
private int getChargingCost(int startHour, int chargeTime)
{
  int cost = 0;
  for (int x = 0; x < chargeTime; x++)
  {
    cost += this.rateTable[(startHour + x) % 24];
  }
  return cost;
}
```

PART B:

```
/** Determines start time to charge the battery at the lowest
 *    cost for the given charge time.
 *  @param chargeTime the number of hours the battery needs to be charged
 *         Precondition: chargeTime > 0
 *  @return an optimal start time, with 0 ≤ returned value ≤ 23
 */
public int getChargeStartTime(int chargeTime)
{
  int startTime = 0;
  for (int i = 1; i < 24; i++)
  {
    if (this.getChargingCost(i, chargeTime)
        < this.getChargingCost(startTime, chargeTime))
    {
      startTime = i;
    }
  }
  return startTime;
}
```

These canonical solutions serve an expository role, depicting general approaches to a solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.

AP® COMPUTER SCIENCE A
2009 CANONICAL SOLUTIONS

Question 4: Tile Game

PART A:

```
/** Determines where to insert tile,
 *  in its current orientation, into game board
 *  @param tile the tile to be placed on the game board
 *  @return the position of tile where tile is to be inserted:
 *        0 if the board is empty;
 *       -1 if tile does not fit in front, at end,
 *            or between any existing tiles;
 *        otherwise, 0 ≤ position returned ≤ board.size()
 */
private int getIndexForFit(NumberTile tile)
{
  if ((this.board.size() == 0) ||
      (tile.getRight() == this.board.get(0).getLeft()))
    return 0;
  for (int i = 1; i < this.board.size(); i++)
  {
    if (tile.getLeft() == this.board.get(i-1).getRight() &&
        tile.getRight() == this.board.get(i).getLeft())
      return i;
  }
  if (tile.getLeft() == this.board.get(this.board.size() - 1).getRight())
    return this.board.size();

  return -1;
}
```

These canonical solutions serve an expository role, depicting general approaches to a solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.

AP® COMPUTER SCIENCE A
2009 CANONICAL SOLUTIONS

Question 4: Tile Game (continued)

PART B:

```
/** Places tile on the game board if it fits (checking all possible
 *    tile orientations if necessary).
 *  If there are no tiles on the game board,
 *    the tile is placed at position 0.
 *  The tile should be placed at most 1 time.
 *  Precondition: board is not null
 *  @param tile the tile to be placed on the game board
 *  @return true if tile is placed successfully; false otherwise
 *  Postcondition: the orientations of the other tiles on the board
 *                 are not changed
 *  Postcondition: the order of the other tiles on the board
 *                 relative to each other is not changed
 */
public boolean insertTile(NumberTile tile)
{
  int index = getIndexForFit(tile);
  int test = 1;
  while (index == -1 && test < 4)
  {
    tile.rotate();
    index = getIndexForFit(tile);
    test++;
  }
  if (index != -1)
    this.board.add(index, tile);

  return (index != -1);
}
```

These canonical solutions serve an expository role, depicting general approaches to a solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.