AP® COMPUTER SCIENCE A
2013 SCORING GUIDELINES

Question 2: TokenPass

Part (a)	`TokenPass` constructor	**4 points**

Intent: *Create* `TokenPass` *object and correctly initialize game state*

+1 Creates instance variable `board` as `int` array of size `playerCount`

+1 Computes a random number between 1 and 10, inclusive, and a random number between 0 and `playerCount-1`, inclusive

+1 Initializes all entries in `board` with computed random value *(no bounds errors)*

+1 Initializes instance variable `currentPlayer` to computed random value

Part (b)	`distributeCurrentPlayerTokens`	**5 points**

Intent: *Distribute all tokens from* `currentPlayer` *position to subsequent positions in array*

+1 Uses initial value of `board[currentPlayer]` to control distribution of tokens

+1 Increases at least one `board` entry in the context of a loop

+1 Starts distribution of tokens at correct board entry

+1 Distributes next token (if any remain) to position 0 after distributing to highest position in board

+1 On exit: token count at each position in `board` is correct

Question-Specific Penalties

-2 (v) Consistently uses incorrect array name instead of `board`

-1 (y) Destruction of persistent data (`currentPlayer`)

-1 (z) Attempts to return a value from `distributeCurrentPlayerTokens`

2Aa

(a) Write the constructor for the `TokenPass` class. The parameter `playerCount` represents the number of players in the game. The constructor should create the `board` array to contain `playerCount` elements and fill the array with random numbers between 1 and 10, inclusive. The constructor should also initialize the instance variable `currentPlayer` to a random number between 0 and `playerCount-1`, inclusive.

Complete the `TokenPass` constructor below.

```
/** Creates the board array to be of size playerCount and fills it with
 *  random integer values from 1 to 10, inclusive. Initializes currentPlayer to a
 *  random integer value in the range between 0 and playerCount-1, inclusive.
 *  @param playerCount the number of players
 */
public TokenPass(int playerCount)
```

```
public TokenPass (int playerCount){
    board = new int[playerCount];
    for(int i=0; i<playerCount; i++){
        board[i] = (int)(Math.Random() * 10) +1;
    }
    currentPlayer = (int)(Math.Random() * playerCount);
}
```

Part (b) begins on page 14.

GO ON TO THE NEXT PAGE.

2Ab

ADDITIONAL WORK SPACE

```
public void distributeCurrentPlayerTokens () {
    int n = board[currentPlayer];
    board[currentPlayer] = 0;
    for (int i = currentPlayer +1; i < board.length; i++){
        if(n > 0){
            board[i] = board[i] + 1;
            n--;
        }
    }
    while( n >0){
        for( int i=0 ; i <board.length ; i++){
            if (n >0){
                board[i] = board[i] +1;
                n--;
            }
        }
    }
}
```

GO ON TO THE NEXT PAGE.

2Ba

(a) Write the constructor for the `TokenPass` class. The parameter `playerCount` represents the number of players in the game. The constructor should create the `board` array to contain `playerCount` elements and fill the array with random numbers between 1 and 10, inclusive. The constructor should also initialize the instance variable `currentPlayer` to a random number between 0 and `playerCount-1`, inclusive.

Complete the `TokenPass` constructor below.

```
/** Creates the board array to be of size playerCount and fills it with
 *   random integer values from 1 to 10, inclusive. Initializes currentPlayer to a
 *   random integer value in the range between 0 and playerCount-1, inclusive.
 *   @param playerCount the number of players
 */
public TokenPass(int playerCount)
```

```
{
    Current player = (int) ( Math.Random * playerCount );
     board = new int [playerCount];
     for ( int i = 0; i < playerCount - 1; i++)
     {
         board [i] = (int) ( Math.Random() * 10 ) + 1;
     }
}
```

Part (b) begins on page 14.

GO ON TO THE NEXT PAGE.

2Bb

(b) Write the `distributeCurrentPlayerTokens` method.

The tokens are collected and removed from the game board at the current player's position. These tokens are distributed, one at a time, to each player, beginning with the next higher position, until there are no more tokens to distribute.

Class information repeated from the beginning of the question

```
public class TokenPass

private int[] board
private int currentPlayer
public TokenPass(int playerCount)
public void distributeCurrentPlayerTokens()
```

Complete method `distributeCurrentPlayerTokens` below.

```
/** Distributes the tokens from the current player's position one at a time to each player in
 *  the game. Distribution begins with the next position and continues until all the tokens
 *  have been distributed. If there are still tokens to distribute when the player at the
 *  highest position is reached, the next token will be distributed to the player at position 0.
 *  Precondition: the current player has at least one token.
 *  Postcondition: the current player has not changed.
 */
public void distributeCurrentPlayerTokens()
{
    int tokens= board[currentPlayer];
    int count = currentPlayer;
    for ( int i=tokens; i > 0; i -- )
    {
        if ( count == board.length - 1 )
        {
            count=0;
            board[count] = board[count]+1;
        }
        else
        {
            board[count+1]= board[count+1]+1;
            count ++;
        }
    }
```

GO ON TO THE NEXT PAGE.

2Ca

(a) Write the constructor for the `TokenPass` class. The parameter `playerCount` represents the number of players in the game. The constructor should create the `board` array to contain `playerCount` elements and fill the array with random numbers between 1 and 10, inclusive. The constructor should also initialize the instance variable `currentPlayer` to a random number between 0 and `playerCount-1`, inclusive.

Complete the `TokenPass` constructor below.

```
/** Creates the board array to be of size playerCount and fills it with
 *  random integer values from 1 to 10, inclusive. Initializes currentPlayer to a
 *  random integer value in the range between 0 and playerCount-1, inclusive.
 *  @param playerCount the number of players
 */
public TokenPass(int playerCount)
{
      board [ ] = new board (playerCount);
      for ( int num : board)
         { num = (int)(Math.random()* 10 + 1);
         }
      currentPlayer = (Math.random()* playerCount);
}
```

Part (b) begins on page 14.

GO ON TO THE NEXT PAGE.

2Cb

(b) Write the `distributeCurrentPlayerTokens` method.

The tokens are collected and removed from the game board at the current player's position. These tokens are distributed, one at a time, to each player, beginning with the next higher position, until there are no more tokens to distribute.

Class information repeated from the beginning of the question

```
public class TokenPass

private int[] board
private int currentPlayer
public TokenPass(int playerCount)
public void distributeCurrentPlayerTokens()
```

Complete method `distributeCurrentPlayerTokens` below.

```
/** Distributes the tokens from the current player's position one at a time to each player in
 *  the game. Distribution begins with the next position and continues until all the tokens
 *  have been distributed. If there are still tokens to distribute when the player at the
 *  highest position is reached, the next token will be distributed to the player at position 0.
 *  Precondition: the current player has at least one token.
 *  Postcondition: the current player has not changed.
 */
public void distributeCurrentPlayerTokens()
{
    int token = board[currentPlayer];
    int x = currentplayer;
    int y = board.length;
    for (k=0; k<=token; k++)
    {
      if (currentplaye>= y)
          {x -= playercount;
          }
      board[x+1]++;
        x++;
      currentplayer ++;

}
```

GO ON TO THE NEXT PAGE.

AP® COMPUTER SCIENCE A
2013 SCORING COMMENTARY

Question 2

Overview

This question involved use of the array data structure, array traversal, and both access and modification of array elements. Students were asked to implement a constructor and a method of the `TokenPass` class. In part (a), students were required to implement the constructor, which creates an `int` array of length `playerCount` (the constructor parameter) and initializes each element in that array to a random integer between1 and 10, inclusive. In addition, the `currentPlayer` must be initialized to a random variable between 0 and `playerCount`. In part (b), students were required to implement the method `distributeTokens`, which distributes the tokens at `board[currentPlayer]`, one at a time, to subsequent positions in the array. If tokens remain to be distributed after the last board position is updated, distribution must continue at position 0. Distribution stops when `board[currentPlayer]` tokens have been distributed.

Sample: 2A
Score: 9

In part (a), an array of `playerCount` integers is correctly created and assigned to the instance variable `board`. Random integer values between 1 and 10, inclusive, are correctly assigned to each value in the `board` instance variable. A random value between 0 and `playerCount - 1`, inclusive, is correctly assigned to the instance variable `currentPlayer`. Part (a) earned 4 points.

In part (b), the number of tokens to distribute is correctly determined. The method distributes tokens to all the players after the current player. The method correctly starts at `currentPlayer + 1` and does not have the possibility of an out-of-bounds error. The method ensures there is always a token available before distributing it. After distributing tokens to all the players after the current one, another pair of loops is used to distribute to all players, beginning with the first. The method ensures there is a token before distributing it. Part (b) earned 5 points.

Sample: 2B
Score: 7

In part (a), a random value between 0 and `playerCount - 1`, inclusive, is correctly assigned to the instance variable `currentPlayer`. An array of `playerCount` integers is correctly created and assigned to the instance variable `board`. Random integer values between 1 and 10, inclusive, are assigned to all values in the `board` instance variable except the last value. Because a random value is not assigned to the last value, the solution did not receive a point for `board` initialization. Part (a) earned 3 points.

In part (b), the method correctly retrieves the number of tokens the current player has. The method does not reset the current player's count to 0, so the current player will end up with more tokens than they should have. This is penalized in the last ("all correct") point only. The method then correctly sets the variable `count to currentPlayer` and correctly increments `count` before using it, so the solution receives a point for starting at the correct location. At the beginning of the loop, the `count` variable is correctly checked to ensure it is in bounds and the position is correctly incremented, so the solution receives a point for wrapping correctly. Elements of `board` are incremented correctly. Part (b) earned 4 points.

Question 2 (continued)

Sample: 2C
Score: 3

In part (a), an array is created incorrectly because it is created as an array of multiple `board` values instead of multiple `int` values. Random integer values between 1 and 10 inclusive are created; an attempt is made to assign these random values to each value in the `board` instance variable. Unfortunately, the method uses an enhanced `for` (for-each) loop, which does not copy changes to the loop control variable to the underlying array. A random value between 0 and `playerCount - 1`, inclusive, is assigned to the instance variable `currentPlayer`. This value is not cast to an `int` so, although the solution did receive the point for initializing `currentPlayer`, the point for random numbers is lost because the method does not correctly create all the random numbers needed. Part (a) earned 1 point.

In part (b), the method correctly retrieves the number of tokens the current player has. The method does not reset the current player's count to 0, so the current player will end up with more tokens than they should have. This is penalized in the last ("all correct") point only. The method then loops through the number of tokens, but because the loop is from 0 to `token`, inclusive, it will loop one too many times. This is also penalized in the last ("all correct") point only. The method does increment board elements and begins the distribution at the correct location. Rather than create a separate variable to step through the board, the method uses `currentPlayer`. Because `currentPlayer` is not reset, once `currentPlayer` equals `y, x` will be decremented on each subsequent iteration of the loop. Additionally, `playerCount` is used for the reset and there is no `playerCount` in scope in this method. For either of these reasons, the player does not correctly wrap. Because `currentPlayer` is changed, there is an additional 1 point penalty for violating post-conditions and changing persistent data. Part (b) earned 2 points.