CS305J Spring 2010

Final Key, Suggested Solutions, and Grading Criteria

Abbreviations:
NAP - no answer provided
ECF - error carried forward
OBOE - off by one error
BOD - benefit of the doubt
GCE - misunderstood question. Answer is way off base.

1. Answer as shown or -1. If mistake on data type, error carried forward to next answers.

```
1. 25
2. 8
3. "4IT8"  (-1 if no quotes. ECF)
4. 0
5. "SC15"  (ECF possible)
6. true    (differences in capitalization okay)
7. 99
8. true    (differences in capitalization okay)
9. 2.5
10. false  (differences in capitalization okay)
```

2.
0.5 point each. No error carried forward.

	`result == -1`	`low <= high`	`list[mid] == tgt`	`low == 0`
`Point A`	A	S	S	A
`Point B`	A	A	S	S
`Point C`	N	A	A	S
`Point D`	S	S	S	S
`Point E`	S	S	S	S

Point A, low <= high -> what if size = 0? low = 0, high = -1

3. Suggested Solution:

```
    public static int getTicketPrice(int basePrice, char day, boolean nonStop,
            boolean firstClass) {
        if(nonStop && (day == 'M' || day == 'F'))
            basePrice *= 2;
        else if(nonStop)
            basePrice += basePrice * 40 / 100;
        else if( day == 'M' || day == 'F' )
            basePrice += basePrice * 30 / 100;
        if(firstClass)
            basePrice += basePrice * 3 / 2;
        return basePrice;
    }
```

Criteria:
10 points. -1 possible for math errors or logic errors

- handle if nonstop only, 2 points
- handle if Monday or Friday only, 2 points
- handle if both nonstop and (Monday or Friday), 2 points
- handle first class, 2 points
- return answer, 2 points

4. Suggested Solution:

```
public static String infrequentCharacters(int[] freqMap) {
     String result = "";
     int totalChar = 0;
     for(int i = 0; i < freqMap.length; i++)
         totalChar += freqMap[i];
     double twoPercentTotal = totalChar * 0.02;
     for(int i = 0; i < freqMap.length; i++)
         if(freqMap[i] < twoPercentTotal)
             result += (char)(i + 'a');
     return result;
 }
```

Criteria:
12 points. partial deductions possible for logic errors

- Create resulting String, 1 point
- calculate total number of chars, attempt 1 point, correct 3 points
- calculate 2% of total (okay if recalculate, no need to guard dived 0), 2 points
- add all infrequent chars to result, attempt 1 point, correct 3 points
- return result, 1 point

5. Suggested Solution

```
public static ArrayList<String> removeWords(ArrayList<String> org,
                                    int[] freqMap) {
    String infrequentChars = infrequentCharacters(freqMap);
    ArrayList<String> allInfrequent = new ArrayList<String>();
    int i = 0;
    while( i < org.size() ){
        String temp = org.get(i).toLowerCase();
        boolean allInfreq = true;
        int indexInString = 0;
        while(allInfreq && indexInString < temp.length()){
            allInfreq =
                infrequentChars.indexOf(temp.charAt(indexInString)) != -1;
            indexInString++;
        }
        if(allInfreq)
            allInfrequent.add(org.remove(i));
        else
            i++;
    }
    return allInfrequent;
}
```

Criteria:
13 points. partial deductions possible for logic errors

Create resulting ArrayList, 1 point

obtain String by calling infrequentCharaters method, 1 point

loop through elements of org,
- attempt 1 point
- correct 2 points (-2 if remove bug)

for current element in org check if all characters are infrequent
- attempt 2 points
- correct 3 points

add Strings that are all infrequent chars to result, 1 point

add Strings that are all infrequent chars from org, 1 point

return result, 1 point

6. Suggested Solution:

```
public class FlyTrap implements Critter{

    private Color myColor;
    private int myDirection;
    private String myChar;

    public FlyTrap(){
        myChar = "F";

        if(Math.random() < 0.8)
            myColor = Color.GREEN;
        else
            myColor = Color.RED;

        double randForDir = Math.random();
        if(randForDir < 0.5)
            myDirection = NORTH;
        else if(randForDir < 0.75)
            myDirection = EAST;
        else
            myDirection = WEST;
    }

    public Color getColor(){
        return myColor;
    }

    public int getMove(CritterInfo info){
        if(Math.random() < 0.75)
            return CENTER;
        else
            return myDirection;
    }

    public String toString(){
            return myChar;
    }

    public int fight(String opponent){
        if(myColor == Color.GREEN){
            myChar = "S";
            return SCISSORS;
        }
        else if(Math.random() < 0.5){
            myChar = "R";
            return ROCK;
        }
        else{
            myChar = "P";
            return PAPER;
        }
    }
}
```

Instance vars correct: 3 points
Constructor correct: 5 points
getColor method correct: 1 point
getMove method correct: 3 points
toString method correct: 1 point
fight method correct: 4 points

7. Suggested Solution

```
private static void flipBoard(char[][] board)
{
   for(int c = 0; c < board[0].length; c++)
   {
       int numEmpty = findNumEmpty(c, board);
       int numToFlip = board.length - numEmpty;
       for(int offset = 0; offset < numToFlip / 2; offset++)
       {
           int topRow = numEmpty + offset;
           int bottomRow = board.length - 1 - offset;
           char temp = board[topRow][c];
           board[topRow][c] = board[bottomRow][c];
           board[bottomRow][c] = temp;
       }
   }
}

private static int findNumEmpty(int col, char[][] board)
{
    int row = 0;
    while(row < board.length && board[row][col] == '.')
        row++;
    return row;
}
```

Process each column:
- attempt, 2 points
- correct, 3 points

Find first checker in column:
- attempt, 2 points
- correct, 3 points

Flip checkers in column:
- attempt, 2 points
- correct, 3 points

Make changes to board parameter, 3. (cannot use new 2d array unless copy it all back into board and cannot return 2d array, method was void.)

8. Suggested Solution

```
public static void runSim(int typeA, int typeB){
    int dayNum = 0;
    System.out.print("Day " + dayNum + ". Population of each type, A and B: ");
    System.out.println(typeA + " " + typeB);
    while(typeB > 0){
        // check if all type B can eat. If not reduce typeB
        if(typeA >= typeB * 3)
            typeA -= typeB * 3;
        else{
            typeB = typeA / 3;
            typeA = typeA % 3;
        }
        typeB += typeB / 10;
        typeA += typeA / 2;
        dayNum++;
        System.out.print("Day " + dayNum + ". Pop of each type, A and B: ");
        System.out.println(typeA + " " + typeB);
    }
}
```

while loop: 3 points

code for typeA eaten:
- attempt: 2 points
- correct: 3 points

new typeA calculated: 1 point
new typeB calculated: 1 point

Track day number: 1 point

print correct data: 1 point