COP 3503 Spring 2022 Section 1 Exam #1 Solution

Java API

1) (10 pts) Complete the program below so that it reads in an integer, ***n*** from standard input, and then reads in ***n*** strings, representing people, coming in order to the DMV. When a person comes for the first time, they should be assigned the first unassigned positive integer ID, and the program should print out, "Giving ***s*** id number ***d***.", where ***s*** is the name of the person. If a person comes for a repeat visit, the program should print out, "Person with id number ***d*** visits again." Use a **HashMap** to complete this program efficiently. (Nearly all the credit will be given for the appropriate use of a HashMap.)

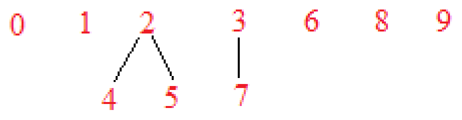
```
import java.util.*;

public class personlist {

    public static void main(String[] args) {

        Scanner stdin = new Scanner(System.in);
        int n = stdin.nextInt();
        HashMap<String,Integer> map = new HashMap<String,Integer>();
        int id = 1;

        for (int i=0; i<n; i++) {

            String name = stdin.next();

            if ( map.containsKey(name) ) {     // 3 pts

                System.out.print("Person with id number ");

                System.out.println( map.get(name) +" visits again."); // 3 pts
            }

            else {

                System.out.println("Giving "+name+" id number "+id);

                map.put(name, id) ;      // 3 pts

                id++ ; // 1 pt

            }
        }
    }
}
```

Disjoint Sets

2) (15 pts) Consider a disjoint set of 10 elements, numbered 0 through 9, where the root of larger value always gets attached to the root of smaller value in union operations, and there is no path compression. Draw the trees, clearly indicating the root (to be at the top), after each of the following operations has been executed on a newly created disjoint set of 10 separate sets. For grading purposes, show the picture after each of the labeled points in the steps (A, B, C)

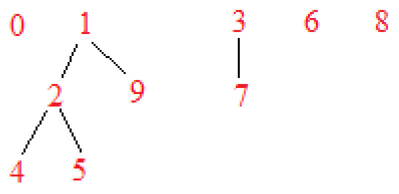
```
union(3, 7)
union(2, 5)
union(4, 5)
----------------- Point A
union(1, 9)
union(4, 9)
----------------- Point B
union(8, 0)
union(5, 6)
union(2, 8)
----------------- Point C
```

Picture after Point A



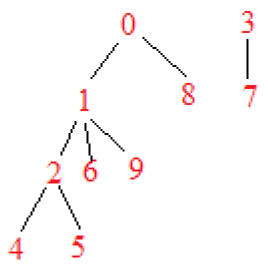
Grading: 2 pts for 2,4,5 config (4 5 can be either order), 1 pt for 3,7 config, 2 pts for rest separate.

Picture after Point B



Grading: 2 pts to attach (2,4,5) to 1, 2 pts to attach 9 directly to 1, 1 pt for keeping rest the same.

Picture after Point C



Grading: 2 pts 6 to 1, 1 ot 8 to 8, 2 pts 1 to 0.

3) (25 pts) Consider the problem of being given a permutation of the integers 1 through n, with the commas in between terms removed. For example, for n = 13, the permutation 3,11,8,1,2,13,7,12,10,6,9,5,4 would appear as 31181213712106954. For this problem, complete a program that reads in the value of n and the String of a permutation (with the commas removed), and prints out a valid way of inserting the commas to produce a permutation. (Notice that the answer isn't always unique. For the above example, we can exchange where 1 and 2 are with 12 to create a second valid answer.) n is limited to be in between 1 and 50, inclusive.

The strategy to solve the problem is as follows: the recursive function that does backtracking takes in two important integers: ***k***, the number of terms in the permutation that have been fixed and ***strIdx***, the current index into the string where the start of the kth term will be selected. (You'll recognize and understand what the other two parameters are…) The goal of the function is to return true if there is a solution with the first ***k*** items fixed. To do this, there are only 2 possible items that could be the (***k***+1)st item: either the one digit integer at index ***strIdx***, OR the two digit integer using the digits at both ***strIdx*** AND ***strIdx***+1. Complete code on the back of this page to solve the problem. All of the set up code is on this side and the method you need to complete is on the back side of this page.

```
import java.util.*;

public class whichperm {

      public static int n;
      public static String input;

      public static void main(String[] args) {

            Scanner stdin = new Scanner(System.in);
            n = stdin.nextInt();
            input = stdin.next();

            int[] perm = new int[n];
            boolean[] used = new boolean[n+1];
            boolean res = go(perm, used, 0, 0);

            for (int i=0; i<n-1; i++)
                  System.out.print(perm[i]+", ");
            System.out.println(perm[n-1]);
      }
public static boolean go(int[] perm, boolean[] used,
                              int k, int strIdx) {

           if (k == n) return true;

           // Calculate the one digit option here.

           int oneD = input.charAt(strIdx) - '0'; // 3 pts

           if (oneD == 0) return false;           // 1 pt

           if (oneD <= n && !used[oneD]) {

                used[oneD] = true;                // 1 pt

                perm[k] = oneD;                   // 2 pts
                boolean tmp = go(perm, used, k+1, strIdx+1);// 2 pts

                if (tmp) return true;             // 1 pt

                used[oneD] = false;               // 1 pt
           }

           if (strIdx+1 == input.length()) return false;

           // Calculate the two digit option here.

           int twoD = 10*(oneD) + (input.charAt(strIdx+1)-'0'); //5pts

           if (twoD <= n && !used[twoD]) {

                used[twoD] = true;                          // 1 pt

                perm[k] = twoD;                             // 2 pts

                boolean tmp = go(perm, used, k+1, strIdx+2); // 3 pts

                if (tmp) return true;                       // 1 pt

                used[twoD] = false;                         // 1 pt
           }

           return false;                                    // 1 pt
      }
}
```

2-4 Trees

4) (5 pts) Show the result of inserting 24 into the 2-4 Tree shown below:

```
              6, 15
      /         |          \
2, 4       8, 11,13     22, 27, 30
```

```
              6, 15, 27
      /        /      \              \
2, 4    8, 11,13  22, 24        30
```

Grading: 2 pts for splitting 27 to root, 2 pts for having 22,24 in 3rd leaf, and 1 pt for having 30 in last leaf.

5) (5 pts) Show the result of deleting 11 from the 2-4 Tree shown below:

```
              6, 15
      /         |          \
2, 4          11          30
```

```
              4, 15
      /         |          \
2               6          30
```

Grading: 2 pts for moving 4 to root, 3 pts for moving 6 to second left.

6) (5 pts) Show the result of deleting 70 from the 2-4 Tree shown below:

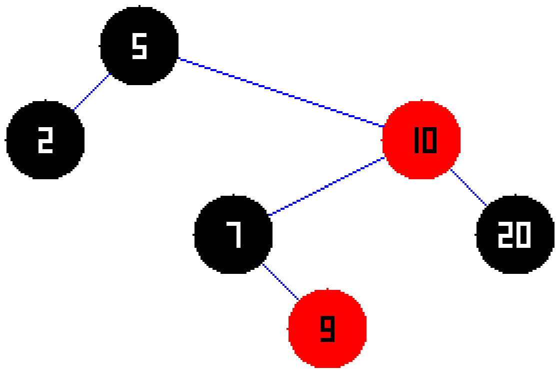
```
             30
          /       \
        10         50
      /    \     /    \
  2,6     20   40   70
```

```
            10  , 30
          /       |       \
      2,6       20      40, 50
```

Grading: 2 pts for 50 to drop in with 40, 2 pts for 30 to drop in with 10, 1 pt for arrangement of leaves.

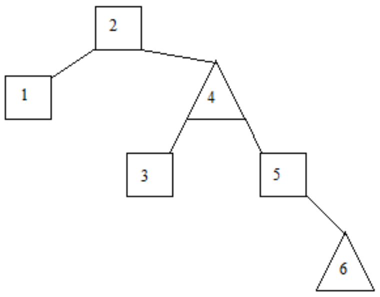
Red-Black Trees

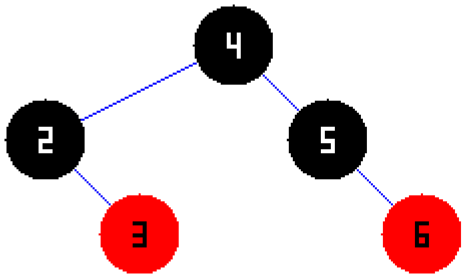
7) (5 pts) Show the end picture of inserting the following nodes into an initially empty Red-Black Tree, in this order: 10, 5, 2, 20, 7, 9. To show black nodes, draw a square the number in the node and to show red nodes, draw a triangle around the number in the node.



Grading: 1 pt 5 at root, 1 pt 2 to left, 1 pt for arrangements of 7,9,10,20, 2 pts for all the colors.

8) (5 pts) Show the result of deleting the value of 1 from the Red-Black Tree shown below. Black nodes have squares around them and red nodes have triangles around them.





Grading: 1 pt 4 at root, 1 pt 2-3 subtree, 1 pt 5-6 subtree, 2 pts all colors

9) (5 pts) One clever solution to the CD problem (Kattis Problem #1) involved just storing one HashSet of both of Jack and Jill's CDs and then subtracting the size of that set from the sum of the total number of CDs that Jack had and the total number of CDs that Jill had. An expression for the answer printed was:

```
numJill+numJack-myset.size()
```

where numJill was the # of CDs Jill had, numJack was the number of CDs Jack had and myset was the HashSet for which each of Jack and Jill's CDs were added.

Explain ***why*** this solution works.

This formula is making use of the inclusion exclusion principle. When we subtract out the cardinality of the union of two sets from the sum of the cardinalities of the two sets, we get the size of the intersection of two sets. Intuitively, when we add the number of CDs from both Jack and Jill, each CD that is owned by both is counted twice and each unique CD (not owned by both) is counted once. When we subtract out the union of the two sets, we are subtracting out one copy of each CD, leaving only one copy of each CD which was originally a duplicate when they put their CDs together.

Grading: Inclusion-Exclusion doesn't have to be mentioned by name, and mentioning it without any explanation does not get full credit. Give partial credit as you see fit, depending on the clarity of the response. Many intuitive explanations of the principle are clear and easy to understand, any of these get full credit. Naming the principle without any explanation is worth 3 pts out of 5.

10) (5 pts) In one of the politics solutions and in both of the Tentaizu solutions posted, a technique was used to store an ordered pair of ints into a single int (or long). Consider using this same system to store an ordered pair (x, y) where $0 \leq x < 250$ and $0 \leq y < 500$. Give the single integer that would correspond to the ordered pair (40, 327). Let x represent the "row" and y represent the "column" in the conversion and make sure that all ordered pairs map to integers in the range [0, 124999]. (Note: The values are chosen so that the arithmetic by hand is relatively easy.)

Answer = 40 x 500 + 327 = 20000 + 327 = **20,327**

Also accepted: 327 x 250 + 40 = 324 x 250 + 3 x 250 + 40
= 81 x (4 x 250) + 750 + 40
= 81 x 1000 + 790
= **81,790**

Grading: 3 pts for multiplying either 40 by 500 or 324 by 250 (0 pts for any other product), 1 pt for adding the opposite coordinate, 1 pt for arithmetic. (0 pts for 40 x 327, 1 pt for 40,327, 4 pts for using bitwise shift of 9 instead of 500, 3 pts if used shift and incorrect arithmetic, 0 pts if left in bits)

11) (5 pts) Recall that $n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots = 2n$. How does this mathematical statement relate to the expected number of physical nodes in a Skip List (original version shown in class before we started tinkering with it) that contains n different values?

The i^{th} term in the sum (starting with i=0) represents the expected number of nodes on level i. Explicitly, there are n nodes on level 0, and then we expect about n/2 nodes on level 1, about n/4 nodes on level 2. So each item on the left represents the expected number of nodes on each level. Thus, the expected total number of nodes in a skip list is simply the sum at all levels. This means that on average, we expect to store 2n nodes for a skip list storing n values and the space complexity of a list of n values is O(n).

Grading: 2 pts for saying something about the halving behavior of the list, 2 pts for clearly stating that each item in the sum represents the expected number of nodes at each different level of the list, 1 pt for tying it together.

12) (5 pts) In the posted solution, tentaizu_alt.java, DX/DY arrays were used. Explain why using these arrays inside of a loop is better than using an if statement with many branches.

By putting all the movement constants in one place (top constant arrays), it's much easier to debug than a bunch of if statements where each set of potentially 8 pairs of constants is typed manually in multiple places. In the former, we just have to make sure our values are correct in one place all next to each other visually. In the latter, our bounds check might have a typo or our array index might have one, so twice as many places (at a minimum) for there to be an error.

Also, the code with DX/DY arrays is much more compact, so it's easier to manage and find a bug in due to its size. Finally, this code encapsulates general logic cleanly (one case) instead of having to separate out the work into potentially 8 cases, which for some students represents typing 8 separate recursive calls instead of one.

Grading: 0 pts for stating it's faster (it's not), 2 or 3 points for stating that the code is cleaner (depends on how they describe this), final 2 pts for explaining why it's easier to debug. 3 pts for explaining how to use the DX/DY arrays without stating why it's better.

13) (5 pts) One of the featured events in the Winter Olympics is snowboarding. On top of what form of precipitation does a person typically ride a snowboard?

Snow (Grading: give to all)