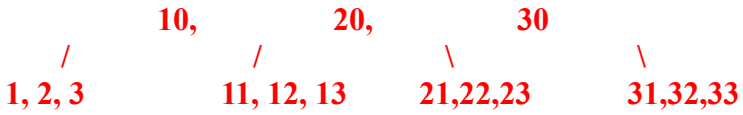
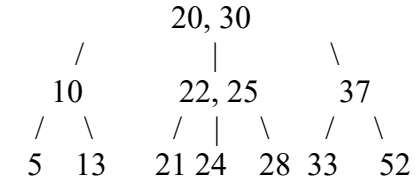
Spring 2024 COP 3503 Exam #1 2/6/2024 – Part C (DJ Sets, 2-4 Trees) Solution

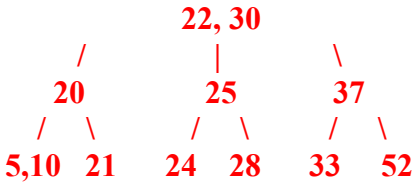
7) (5 pts) Draw a valid 2-4 Tree of height 1 with 15 distinct positive integers in it.



Grading: 1 pt valid 2-4 tree of height 1 (with inequalities satisfied)
1 pt for root node having 3 values
3 pts for the rest

8) (5 pts) Show the result of deleting 13 from the 2-4 Tree shown below:





Note: the 10 drops into the vacated nodewhere 13 used to be and then fuses with 5. Then, a transfer operation occurs with 22 as it takes the place of 20 and 20 drops into the spot vacated by 10. Then the 21 reattaches itself where it has to go.

Grading: 1 pt for leaving 33,37,52 unchanged
2 pts for having 22, 30 in root
2 pts for fixing everything else accordingly.

9) (15 pts) One application of a disjoint set is to mark the number of connected regions on a 2D grid. Consider an input grid of upper case letters. A connected region is one which consists of the same letter which share an edge on the grid (so a square that is up, down, left or right from another is connected to it. For example, in the grid below there are 5 connected regions (2 marked A, 1 marked B, C and D.)

A	A	D	B	A
A	A	D	B	B
A	D	D	C	C
A	A	D	D	C

Complete the code on the next page so that it reads in this grid and outputs the # of connected regions in the grid. **(Note: Disjoint Set Code at end of Part D.)**

```
import java.util.*;

public class connected {

    public static int r;
    public static int c;
    public static char[][] g;

    final public static int[] DX = {0,1};
    final public static int[] DY = {1,0};

    public static void main(String[] args) {

        Scanner stdin = new Scanner(System.in);
        r = stdin.nextInt();
        c = stdin.nextInt();
        g = new char[r][];
        for (int i=0; i<r; i++)
            g[i] = stdin.next().toCharArray();

        djset dj = new djset(r*c);

        for (int i=0; i<r; i++) {
            for (int j=0; j<c; j++) {
                for (int k=0; k<2; k++) {

                    int nx = i + DX[k];                 // 3 pts
                    int ny = j + DY[k];                 // 3 pts

                    if (nx>=r || ny>=c) continue;       // 2 pts

                    if (g[i][j] == g[nx][ny])           // 2 pts
                        dj.union(i*c+j, nx*c+ny);       // 5 pts



                // end k}
            } // end j
        } // end i

        System.out.println(dj.numSets);
    }
}
```