```
import java.io.File;
import java.util.Scanner;
import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;
import java.io.FileNotFoundException;

public class TylerK_C3Intermediate_Veitch2020 {

    public static void main(String args[]) throws FileNotFoundException {
        //initialization
        Scanner input = new Scanner(new
File("C:\\Users\\tyler\\dev\\TylerKimJavaHomework\\Competition\\src\\TKim_3_ACSLVeit
ch_text"));
        while(input.hasNext()) {
            ArrayList<String> parameters = compressedLocation(input.nextLine());
            ArrayList<ArrayList<String>> locations = new ArrayList<>();
            ArrayList<int[][]> factors = new ArrayList<>();
            int[][] finalArray = create2DArray();
            ArrayList<ArrayList<Integer>> binaryValueSummation = new ArrayList<>();
            ArrayList<String> finalHexadecimalValues = new ArrayList<>();

            //get individual locations data
            for(int i = 0; i < parameters.size(); i++) {
                locations.add(setLocation(parameters.get(i)));
            }

           //create as many 2d arrays as there are in the number of locations
            System.out.println(locations);

           //set number of factors to be number of 2d arrays
            for(int i = 0; i < locations.size(); i++) {
                factors.add(modify2DArray(locations.get(i)));
            }

            //add up all arrays
                //sets the initial value
                int[][] initialArray = developBinary2DArray(factors.get(0),
factors.get(1));

            for(int i = 2; i < factors.size(); i++) {
               finalArray = developBinary2DArray(initialArray, factors.get(i));
               initialArray = finalArray;
            }

            //list all final arrays
            for(int i = 0; i < finalArray.length; i++) {
                ArrayList<Integer> individualBinaryFactors = new ArrayList<>();

                for(int j = 0; j < finalArray.length; j++) {
```

```
                    individualBinaryFactors.add(finalArray[i][j]);
                }
                binaryValueSummation.add(individualBinaryFactors);
            }

            //convert to hexadecimal
            for(int i = 0; i < binaryValueSummation.size(); i++) {

finalHexadecimalValues.add(binaryToHexadecimalConvertor(binaryValueSummation.get(i))
);
            }

            System.out.println(binaryValueSummation);
            System.out.println(finalHexadecimalValues);
        }
    }

    //create compressed grid
    public static ArrayList<String> compressedLocation(String conditions) {
        ArrayList<String> compressedLocationSet = new ArrayList<>();
        String[] tempSet = conditions.split("\\+");

        for(int i = 0; i < tempSet.length; i++) {
            compressedLocationSet.add(tempSet[i]);
        }

        return compressedLocationSet;
    }

    //create grid
    public static int[][] create2DArray() {
        int[][] grid = new int[4][4];
        return grid;
    }

    //get grid of locations
    public static ArrayList<String> setLocation(String compressedLocations) {
        ArrayList<String> location = new ArrayList<>();

        for(int i = 0; i < compressedLocations.length(); i++) {

            if(compressedLocations.substring(i, i+1).equals("~")) {
                location.add(compressedLocations.substring(i, i+2));
                i++;
                continue;

            } else {
                location.add(compressedLocations.substring(i, i +1));
            }
        }
```

```
    return location;
}

//modify 2D array
public static int[][] modify2DArray(ArrayList<String> locations) {
    int[][] final2DArray = create2DArray();

    for(int i = 0; i < locations.size(); i++) {

        //case of ~A set all boxes of A to 0
        if(locations.get(i).equals("~A")) {
            for(int j = 0; j < final2DArray.length; j++) {
                final2DArray[j][0] = 1;
                final2DArray[j][1] = 1;
            }
        }

        //case of A set all boxes of ~A to 0
        if(locations.get(i).equals("A")) {
            for(int j = 0; j < final2DArray.length; j++) {
                final2DArray[j][2] = 1;
                final2DArray[j][3] = 1;
            }
        }

        //case of ~B set all boxes of B to 0
        if(locations.get(i).equals("~B")) {
            for(int j = 0; j < final2DArray.length; j++) {
                final2DArray[0][j] = 1;
                final2DArray[1][j] = 1;
            }
        }

        //case of B
        if(locations.get(i).equals("B")) {
            for(int j = 0; j < final2DArray.length; j++) {
                final2DArray[2][j] = 1;
                final2DArray[3][j] = 1;
            }
        }

        //case of ~C
        if(locations.get(i).equals("~C")) {
            for(int j = 0; j < final2DArray.length; j++) {
                final2DArray[j][1] = 1;
                final2DArray[j][2] = 1;
            }
        }
```

```
            //case of C
            if(locations.get(i).equals("C")) {
                for(int j = 0; j < final2DArray.length; j++) {
                    final2DArray[j][0] = 1;
                    final2DArray[j][3] = 1;
                }
            }

            //case of ~D
            if(locations.get(i).equals("~D")) {
                for(int j = 0; j < final2DArray.length; j++) {
                    final2DArray[1][j] = 1;
                    final2DArray[2][j] = 1;
                }
            }

            //case of D
            if(locations.get(i).equals("D")) {
                for(int j = 0; j < final2DArray.length; j++) {
                    final2DArray[0][j] = 1;
                    final2DArray[3][j] = 1;
                }
            }
        }

        return final2DArray;
    }


    //test if individual blocks are a 1 or zero
    public static int[][] developBinary2DArray(int[][] array1, int[][] array2) {
        int[][] finalArray = create2DArray();

        //test if each individual index is 1 or 0
        for(int i = 0; i < array1.length; i++) {
            for(int j = 0; j < array1.length; j++) {
                if(array1[i][j] == 0 || array2[i][j] == 0) {
                    finalArray[i][j] = 0;
                } else {
                    finalArray[i][j] = 1;
                }
            }
        }

        return finalArray;
    }

    //binary to hexadecimal convertor
    public static String binaryToHexadecimalConvertor(ArrayList<Integer>
binaryValue) {
```

```
        int finalValue = 0;
        String value = null;

        //add up the values
        if(binaryValue.get(0) == 0) {
            finalValue = finalValue + 8;
        }
        if(binaryValue.get(1) == 0) {
            finalValue = finalValue + 4;
        }
        if(binaryValue.get(2) == 0) {
            finalValue = finalValue + 2;
        }
        if(binaryValue.get(3) == 0) {
            finalValue++;
        }

        //result of summation
            //if less than 10
        if(finalValue < 10) {
            value = Integer.toString(finalValue);
        }
            //other values
        switch(finalValue) {
            case 10:
                value = "A";
                break;
            case 11:
                value = "B";
                break;
            case 12:
                value = "C";
                break;
            case 13:
                value = "D";
                break;
            case 14:
                value = "E";
                break;
            case 15:
                value = "F";
                break;
        }

        return value;

    }

}
```