Crossword Puzzle Solution - Java and C++

Java Code

```
import java.io.;
import java.util.;
public class Solution {
private static class Cell {
 int row, col;
 public Cell(int r, int c) {
    this.row = r;
   this.col = c;
   }
}
public static void completeThePuzzle(char[][] puzzle, HashSet
<String> set)
{

    for(int r = 9; r >= 0; r--)
    {
        for(int c = 0; c < 10; c++)
        {
            if(puzzle[r][c] != '+')
            {
               completeThePuzzleHelper(puzzle, set, r, c);
                return;
```

```
            }
        }
    }
}

public static void completeThePuzzleHelper(char[][] puzzle, H
ashSet<String> set, int row, int col)
{
    //System.out.println("row: "+row+" col: "+col);
    //printPuzzle(puzzle);
    //System.out.println("_____________________");
    if(puzzle[row][col] == '-')
    {
        int rCount = getRHSEmptyCellsCount(puzzle, row, col);
        int lStart = getLeftStartCol(puzzle, row, col);
        int rSz = lStart == col ? rCount : rCount+(col-lStart
);
        //System.out.println("row: "+row+" col: "+col+" rSz"+
rSz+" lStart: "+lStart);
        if(rSz > 1)
        {
         for(String s: set)
         {
             if(s.length() == rSz && canRightFit(puzzle, row,
 lStart, s))
             {
                 char[][] puzzleCopy = copyPuzzle(puzzle);
                 HashSet<String> setCopy = copySet(set);
```

```
                    setCopy.remove(s);
                    rightFit(puzzleCopy, row, lStart, s);
                    Cell c = getNextCell(puzzleCopy, row, col);
                   // if(c != null)
                   //     System.out.println("row: "+row+" col:
"+col+" nextRow: "+c.row+" nextCol: "+c.col);
                    if(c == null)
                    {
                       printPuzzle(puzzleCopy);
                       return;
                    }
                    else completeThePuzzleHelper(puzzleCopy, set
Copy, c.row, c.col);
                }
             }
          }
          else
          {
            int dCount = getDownEmptyCellsCount(puzzle, row, co
l);
            int uStart = getUpStartRow(puzzle, row, col);
            int dSz = uStart == row ? dCount : dCount+ uStart-r
ow;
            //System.out.println("row: "+row+" col: "+col+" dSz
: "+dSz+" uStart: "+uStart);
            for(String s: set)
            {
                if(s.length() == dSz && canDownFit(puzzle, uSta
```

```
rt, col, s))
                {
                    char[][] puzzleCopy = copyPuzzle(puzzle);
                    HashSet<String> setCopy = copySet(set);
                    setCopy.remove(s);
                    downFit(puzzleCopy, uStart, col, s);
                    Cell c = getNextCell(puzzleCopy, row, col);
                    // if(c != null)
                    //    System.out.println("row: "+row+" col:
"+col+" nextRow: "+c.row+" nextCol: "+c.col);
                    if(c == null)
                    {
                       printPuzzle(puzzleCopy);
                       return;
                    }
                    else completeThePuzzleHelper(puzzleCopy, set
Copy, c.row, c.col);
                }
            }
          }

     }
    //return false;
}

public static void printPuzzle(char[][] puzzle)
{
     for(int r = 9; r>=0; r--)
```

```
    {
        for(int c = 0; c< 10; c++)
        {
            System.out.print(puzzle[r][c]);
        }
        System.out.println();
    }
}
public static void downFit(char[][] puzzle, int row, int col,
 String s) {
 for(int r = row; r >=0 && row-r+1 <= s.length(); r--)
     puzzle[r][col] = s.charAt(row-r);
}

public static boolean canDownFit(char[][] puzzle, int row, in
t col, String s) {
    for(int r = row; r >=0 && row-r+1 <= s.length(); r--)
        { if(puzzle[r][col] != '-' && puzzle[r][col] != s.cha
rAt(row-r))
          return false;
        }
        return true;
    }
public static Cell getNextCell(char[][] puzzle, int row, int
col)
{
 for(int r = row; r >=0; r--)
 {
```

```
     for(int c = 0; c< 10; c++)
     {
         if(r == row && c >= col && puzzle[r][c] == '-')
             return new Cell(r,c);
         else if (r < row && puzzle[r][c] == '-')
             return new Cell(r,c);
     }
 }
    return null;
}
public static void rightFit(char[][] puzzle, int row, int col
, String s)
{
    for(int c = col; c < 10 && c-col+1 <= s.length(); c++)
        puzzle[row][c] = s.charAt(c-col);
}
public static boolean canRightFit(char[][] puzzle, int row, i
nt col, String s)
{
    for(int c = col; c< 10 && c-col+1 <= s.length(); c++)
    {
        if(puzzle[row][c] != '-' && puzzle[row][c] != s.charA
t(c-col))
            return false;

    }
    return true;
}
```

```
public static int getLeftStartCol(char[][] puzzle, int row, i
nt col)
{

    while(col >=0 && puzzle[row][col] != '+')
    {
        col--;
    }
    return col+1;
}
public static int getUpStartRow(char[][] puzzle, int row, int
 col)
{
    while(row <10 && puzzle[row][col] != '+')
        row++;
    return row-1;
}
public static int getDownEmptyCellsCount(char[][] puzzle, int
 row, int col)
{
    int count = 0;
    while(row >=0 && puzzle[row][col] != '+')
    {
        count++;
        row--;
    }
    return count;
```

```
}
public static int getRHSEmptyCellsCount(char[][] puzzle, int
row, int col)
{

    int count = 0;
    while(col < 10 && puzzle[row][col] != '+')
    {
        count++;
        col++;
    }
    return count;
}
public static HashSet<String> copySet(HashSet<String> set)
{
    HashSet<String> setCopy = new HashSet<>();
    setCopy.addAll(set);
    return setCopy;
}
public static char[][] copyPuzzle(char[][] puzzle)
{
    char[][] puzzleCopy = new char[10][10];
    for(int r = 9; r >=0; r--)
    {
        for(int c = 0; c < 10; c++)
        {
            puzzleCopy[r][c] = puzzle[r][c];
        }
```

```
    }
    return puzzleCopy;
}

public static void main(String[] args) {
    /* Enter your code here. Read input from STDIN. Print out
put to STDOUT. Your class should be named Solution. */
    Scanner sc = new Scanner(System.in);
    char[][] puzzle = new char[10][10];
    for(int r = 9; r >=0; r--)
    {
        String word = sc.nextLine();
        for(int c = 0; c< 10; c++)
            puzzle[r][c] = word.charAt(c);
    }
    String word = sc.nextLine();
    String[] cities = word.split(";");
    HashSet<String> set = new HashSet<>();
    for(String s: cities)
        set.add(s);
    completeThePuzzle(puzzle, set);

}
}
```

C++ Code

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;

bool flag=false;
void print(char a[][10])
    {
    //cout<<endl;
    for(int i=0; i<10; i++)
        {
        for(int j=0; j<10; j++)
            cout<<a[i][j];
        cout<<endl;
    }
}

bool noempty(char a[][10])
    {
    for(int i=0; i<10; i++)
        for(int j=0; j<10; j++)
            if(a[i][j]=='-')
                return false;
    return true;
}
```

```
void fill(char a[][10], string w[], int n, int used, int take
n[10])
{
    if(flag == true)
        return;
    if(used==n && noempty(a))
        {
        flag = true;
        print(a);
    }
    for(int i=0; i<10; i++)
        for(int j=0; j<10; j++)
        {
            if(a[i][j] == '-')
            {
                int h_depth=1;
                int v_depth=1;
                while(j+h_depth<10 && a[i][j+h_depth]!='+')
                    h_depth++;
                while(i+v_depth<10 && a[i+v_depth][j]!='+')
                    v_depth++;
                if(h_depth>1 && (j==0||(j>0 && a[i][j-1] == '
+')))
                {
                    for(int k=0; k<n; k++)
                        if(w[k].length()==h_depth && !taken[k
])
                        {
```

```
                                taken[k] = 1;
                                for(int h=0; h<h_depth; h++)
                                {
                                    if(a[i][j+h]>='A' && a[i][j+h
]<='Z' && a[i][j+h]!=w[k][h])
                                    {
                                        taken[k]=0;
                                        while(h>=0)
                                            a[i][j+h--] = '-';
                                        break;
                                    }
                                    a[i][j+h] = w[k][h];
                                }
                                //cout<<"\nUsed h1"<<w[k]<<i<<j;
                                //print(a);
                                fill(a, w, n, used+1, taken);
                                for(int h=0; h<h_depth; h++)
                                    a[i][j+h] = '-';
                                taken[k] = 0;
                            }
                        }
                        else if(v_depth>1 && (i==0 || (i>0 && a[i-1][
j] == '+')))
                            {
                            for(int k=0; k<n; k++)
                                if(w[k].length()==v_depth && !taken[k
])
                                {
```

```
                                taken[k] = 1;
                                for(int v=0; v<v_depth; v++)
                                {
                                    if(a[i+v][j]>='A' && a[i+v][j
]<='Z' && a[i+v][j]!=w[k][v])
                                    {
                                        taken[k]=0;
                                        while(v>=0)
                                            a[i+v--][j] = '-';
                                        break;
                                    }
                                    a[i+v][j] = w[k][v];
                                }
                                //cout<<"\nUsed v1"<<w[k]<<i<<j;;
                                //print(a);
                                fill(a, w, n, used+1, taken);
                                for(int v=0; v<v_depth; v++)
                                   a[i+v][j] = '-';
                                taken[k] = 0;
                            }
                        }
                        else if(v_depth>1 && (i==0||(i>0 && a[i-1][j]
 != '+')))
                            {
                            for(int k=0; k<n; k++)
                                if(w[k].length()==v_depth+1 && (w[k][
0] == a[i-1][j]) && !taken[k])
                                {
```

```
                                taken[k] = 1;
                                for(int v=0; v<v_depth; v++)
                                {
                                    if(a[i+v][j]>='A' && a[i+v][j
]<='Z' && a[i+v][j]!=w[k][v+1])
                                    {
                                        taken[k]=0;
                                        while(v>0)
                                            a[i+v--][j] = '-';
                                        break;
                                    }
                                    a[i+v][j] = w[k][v+1];
                                }
                                //cout<<"\nUsed v2"<<w[k]<<i<<j;
                                //print(a);
                                fill(a, w, n, used+1, taken);
                                for(int v=0; v<v_depth; v++)
                                    a[i+v][j] = '-';
                                taken[k] = 0;
                            }
                        }

                        else if(h_depth>1 && (j==0 || (j>0 && a[i][j-
1] != '+')))
                            {
                            for(int k=0; k<n; k++)
                                if(w[k].length()==h_depth+1 && (w[k][
0] == a[i][j-1]) && !taken[k])
```

```
                        {
                            taken[k] = 1;
                            for(int h=0; h<h_depth; h++)
                            {
                                if(a[i][j+h]>='A' && a[i][j+h
]<='Z' && a[i][j+h]!=w[k][h+1])
                                {
                                    taken[k]=0;
                                    while(h>0)
                                        a[i][j+h--] = '-';

                                    break;
                                }
                                a[i][j+h] = w[k][h+1];
                            }
                            //cout<<"\nUsed h2"<<w[k]<<i<<j;
                            //print(a);
                            fill(a, w, n, used+1, taken);
                            for(int h=0; h<h_depth; h++)
                                a[i][j+h] = '-';
                            taken[k] = 0;
                        }
                    }
                }
            }
}

int main() {
```

```
    char mat[10][10];
    for(int i=0; i<10; i++)
        for(int j=0; j<10; j++)
            cin>>mat[i][j];
    string s;
    cin>>s;
    string word[10];
    int taken[10] = {0,};
    int counter = 0;
    int p=0;
    flag = false;
    for(int i=0; i<(s.length()); i++)
        if(s[i]==';')
        {
            counter++;
            p=0;
        }
        else
            word[counter] += s[i];

    fill(mat, word, counter+1, 0, taken);
    return 0;
}
```