```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import math
from collections import OrderedDict

def ReadFileAndFillNodes(fileName):
    file=open(fileName,'r')
    row,column = file.readline().split()
    reverseDict = {"N": "S", "S": "N", "W": "E", "E": "W", "NE": "SW", "SW": "NE", "NW": "SE", "SE": "NW", "X": "X"}
    for i in range(int(row)):
        for j in range(int(column)):
            numRow,numColumn,color,circle,direction = file.readline().split()
            # adding normal node
            g.add_node((int(numRow),int(numColumn)),Color=color,Circle=circle,Direction=direction,Parent=None)
            # adding reverse node
            g.add_node((int(numRow),int(numColumn),"R"),Color=color,Circle=circle,Direction=reverseDict[direction],Parent=None)
    file.close()
    return int(row),int(column)

def FillEdges(row,column):
    for i in range(1, row + 1):
        for j in range(1,column+1):
            #going through all the normal nodes
            if g.node[(i,j)]['Direction'] == "N":
                for before in range(1,i):
                    if g.node[(i,j)]['Color'] != g.node[(before,j)]['Color']:
                        #add to edge if the colors don't match (ex. Red -> Blue or Blue -> X)
                        if g.node[(before,j)]['Circle'] == "C":
                            # if one of the vertice is a circle, connect to the reverse node
                            # (et. node on the other layer)
                            g.add_edge((i,j),(before,j,"R"))
                        else:
                            g.add_edge((i,j),(before,j),color='b')
            elif g.node[(i, j)]['Direction'] == "S":
                for after in range(i+1,row+1):
                    if after <= row:
                        if g.node[(i,j)]['Color'] != g.node[(after,j)]['Color']:
                            if g.node[(after,j)]['Circle'] == "C":
                                g.add_edge((i, j), (after, j,"R"))
                            else:
                                g.add_edge((i,j),(after,j))
            elif g.node[(i, j)]['Direction'] == "W":
                for before in range(1,j):
                    if g.node[(i, j)]['Color'] != g.node[(i, before)]['Color']:
                        if g.node[(i,before)]['Circle'] == "C":
                            g.add_edge((i,j),(i,before,"R"))
                        else:
                            g.add_edge((i,j),(i,before))
            elif g.node[(i, j)]['Direction'] == "E":
                for after in range(j+1,column+1):
                    if after <= column:
                        if g.node[(i, j)]['Color'] != g.node[(i,after)]['Color']:
                            if g.node[(i,after)]['Circle'] == "C":
                                g.add_edge((i,j),(i,after,"R"))
                            else:
                                g.add_edge((i,j),(i,after))
            elif g.node[(i, j)]['Direction'] == "NW":
                beforeRow = i-1
                beforeColumn = j-1
                while beforeRow >= 1 and beforeColumn >= 1:
                        if g.node[(i, j)]['Color'] != g.node[(beforeRow,beforeColumn)]['Color']:
                            if g.node[(beforeRow,beforeColumn)]['Circle'] == "C":
                                g.add_edge((i, j), (beforeRow,beforeColumn, "R"))
                            else:
                                g.add_edge((i, j), (beforeRow,beforeColumn))
                        beforeRow-=1
                        beforeColumn-=1
            elif g.node[(i, j)]['Direction'] == "SE":
                afterRow = i+1
                afterColumn= j+1
                while afterRow <= row and afterColumn <= column:
```

```
                    if g.node[(i, j)]['Color'] != g.node[(afterRow, afterColumn)]['Color']:
                        if g.node[(afterRow,afterColumn)]['Circle'] == "C":
                            g.add_edge((i, j), (afterRow, afterColumn, "R"))
                        else:
                            g.add_edge((i, j), (afterRow, afterColumn))
                    afterRow+=1
                    afterColumn+=1
            elif g.node[(i, j)]['Direction'] == "SW":
                beforeRow = i+1
                beforeColumn = j-1
                while beforeRow <= row and beforeColumn >= 1:
                        if g.node[(i, j)]['Color'] != g.node[(beforeRow,beforeColumn)]['Color']:
                            if g.node[(beforeRow, beforeColumn)]['Circle'] == "C":
                                g.add_edge((i, j), (beforeRow, beforeColumn, "R"))
                            else:
                                g.add_edge((i, j), (beforeRow, beforeColumn))
                        beforeRow+=1
                        beforeColumn-=1
            elif g.node[(i, j)]['Direction'] == "NE":
                afterRow = i-1
                afterColumn = j+1
                while afterRow >= 1 and afterColumn <= column:
                        if g.node[(i, j)]['Color'] != g.node[(afterRow, afterColumn)]['Color']:
                            if g.node[(afterRow, afterColumn)]['Circle'] == "C":
                                g.add_edge((i, j), (afterRow, afterColumn, "R"))
                            else:
                                g.add_edge((i, j), (afterRow, afterColumn))
                        afterRow-=1
                        afterColumn+=1

            #going through all the reverse nodes
            if g.node[(i,j,"R")]['Direction'] == "N":
                for before in range(1,i):
                    if g.node[(i,j,"R")]['Color'] != g.node[(before,j,"R")]['Color']:
                        if g.node[(before, j,"R")]['Circle'] == "C":
                            g.add_edge((i, j,"R"), (before, j))
                        else:
                            g.add_edge((i, j,"R"), (before, j,"R"), color='b')
            elif g.node[(i, j,"R")]['Direction'] == "S":
                for after in range(i+1,row+1):
                    if after <= row:
                        if g.node[(i,j,"R")]['Color'] != g.node[(after,j,"R")]['Color']:
                            if g.node[(after, j,"R")]['Circle'] == "C":
                                g.add_edge((i, j,"R"), (after, j))
                            else:
                                g.add_edge((i, j,"R"), (after, j,"R"))

            elif g.node[(i, j,"R")]['Direction'] == "W":
                for before in range(1,j):
                    if g.node[(i, j,"R")]['Color'] != g.node[(i, before,"R")]['Color']:
                        if g.node[(i,before,"R")]['Circle'] == "C":
                            g.add_edge((i,j,"R"),(i,before))
                        else:
                            g.add_edge((i,j,"R"),(i,before,"R"))
            elif g.node[(i, j,"R")]['Direction'] == "E":
                for after in range(j+1,column+1):
                    if after <= column:
                        if g.node[(i, j,"R")]['Color'] != g.node[(i,after,"R")]['Color']:
                            if g.node[(i,after,"R")]['Circle'] == "C":
                                g.add_edge((i,j,"R"),(i,after))
                            else:
                                g.add_edge((i,j,"R"),(i,after,"R"))
            elif g.node[(i, j,"R")]['Direction'] == "NW":
                beforeRow = i-1
                beforeColumn = j-1
                while beforeRow >= 1 and beforeColumn >= 1:
                        if g.node[(i, j,"R")]['Color'] != g.node[(beforeRow,beforeColumn,"R")]['Color']:
                            if g.node[(beforeRow,beforeColumn,"R")]['Circle'] == "C":
                                g.add_edge((i, j,"R"), (beforeRow,beforeColumn))
                            else:
                                g.add_edge((i, j,"R"), (beforeRow,beforeColumn,"R"))
                        beforeRow-=1
                        beforeColumn-=1
            elif g.node[(i, j,"R")]['Direction'] == "SE":
```

```
147                 afterRow = i+1
148                 afterColumn= j+1
149                 while afterRow <= row and afterColumn <= column:
150                     if g.node[(i, j,"R")]['Color'] != g.node[(afterRow, afterColumn,"R")]['Color']:
151                         if g.node[(afterRow, afterColumn,"R")]['Circle'] == "C":
152                             g.add_edge((i, j,"R"), (afterRow, afterColumn))
153                         else:
154                             g.add_edge((i, j,"R"), (afterRow, afterColumn,"R"))
155                     afterRow+=1
156                     afterColumn+=1
157             elif g.node[(i, j,"R")]['Direction'] == "SW":
158                 beforeRow = i+1
159                 beforeColumn = j-1
160                 while beforeRow <= row and beforeColumn >= 1:
161                         if g.node[(i, j,"R")]['Color'] != g.node[(beforeRow,beforeColumn,"R")]['Color']:
162                             if g.node[(beforeRow, beforeColumn,"R")]['Circle'] == "C":
163                                 g.add_edge((i, j,"R"), (beforeRow, beforeColumn))
164                             else:
165                                 g.add_edge((i, j,"R"), (beforeRow, beforeColumn,"R"))
166                         beforeRow+=1
167                         beforeColumn-=1
168             elif g.node[(i, j,"R")]['Direction'] == "NE":
169                 afterRow = i-1
170                 afterColumn = j+1
171                 while afterRow >= 1 and afterColumn <= column:
172                         if g.node[(i, j,"R")]['Color'] != g.node[(afterRow, afterColumn,"R")]['Color']:
173                             if g.node[(afterRow, afterColumn,"R")]['Circle'] == "C":
174                                 g.add_edge((i, j,"R"), (afterRow, afterColumn))
175                             else:
176                                 g.add_edge((i, j,"R"), (afterRow, afterColumn,"R"))
177                         afterRow-=1
178                         afterColumn+=1
179
180 def DisplayResult():
181     dict = {}
182     for parent,child in nx.dfs_tree(g, (1, 1)).edges():
183         dict.update({child:parent})
184
185     DisplayRoute((7,7),dict)
186
187     traceBackList.reverse() #must reverse so that starting point is the first point
188     for x in range(len(traceBackList)):
189         #get rid of all the "R" notations in reverse nodes since that was mainly for the graph traversal
190         if "R" in traceBackList[x]:
191             traceBackList[x] = traceBackList[x][0:5] + ")"
192     for x in traceBackList:
193         print(x,end=" ")
194
195 def DisplayRoute(child,dict): #going through DFS tree until hitting starting point
196     if (child == (1,1)):
197         traceBackList.append(str(child))
198         return
199     else:
200         traceBackList.append(str(child))
201         DisplayRoute(dict[child],dict)
202
203 #main
204 row = 0
205 column = 0
206 traceBackList = []
207 g = nx.DiGraph()
208
209 row,column=ReadFileAndFillNodes("input.txt")
210 FillEdges(row,column)
211 DisplayResult()
212
213 #drawing plot
214 nx.draw(g,with_labels=True)
215 plt.show()
216
217
218
219
220
```