

```

1 import networkx as nx
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5 from collections import OrderedDict
6
7 def ReadFileAndFillNodes(fileName):
8     file=open(fileName,'r')
9     row,column = file.readline().split()
10    reverseDict = {"N": "S", "S": "N", "W": "E", "E": "W", "NE": "SW", "SW": "NE", "NW": "SE", "SE": "NW", "X":
    "X"}
11    for i in range(int(row)):
12        for j in range(int(column)):
13            numRows,numColumns,color,circle,direction = file.readline().split()
14            # adding normal node
15            g.add_node((int(numRow),int(numColumn)),Color=color,Circle=circle,Direction=direction,Parent=None)
16            # adding reverse node
17            g.add_node((int(numRow),int(numColumn),"R"),Color=color,Circle=circle,Direction=reverseDict[
direction],Parent=None)
18    file.close()
19    return int(row),int(column)
20
21 def FillEdges(row,column):
22    for i in range(1, row + 1):
23        for j in range(1,column+1):
24            #going through all the normal nodes
25            if g.node[(i,j)]['Direction'] == "N":
26                for before in range(1,i):
27                    if g.node[(i,j)]['Color'] != g.node[(before,j)]['Color']:
28                        #add to edge if the colors don't match (ex. Red -> Blue or Blue -> X)
29                        if g.node[(before,j)]['Circle'] == "C":
30                            # if one of the vertice is a circle, connect to the reverse node
31                            # (et. node on the other layer)
32                            g.add_edge((i,j),(before,j,"R"))
33                        else:
34                            g.add_edge((i,j),(before,j),color='b')
35            elif g.node[(i, j)]['Direction'] == "S":
36                for after in range(i+1,row+1):
37                    if after <= row:
38                        if g.node[(i,j)]['Color'] != g.node[(after,j)]['Color']:
39                            if g.node[(after,j)]['Circle'] == "C":
40                                g.add_edge((i, j), (after, j,"R"))
41                            else:
42                                g.add_edge((i,j),(after,j))
43            elif g.node[(i, j)]['Direction'] == "W":
44                for before in range(1,j):
45                    if g.node[(i, j)]['Color'] != g.node[(i, before)]['Color']:
46                        if g.node[(i,before)]['Circle'] == "C":
47                            g.add_edge((i,j),(i,before,"R"))
48                        else:
49                            g.add_edge((i,j),(i,before))
50            elif g.node[(i, j)]['Direction'] == "E":
51                for after in range(j+1,column+1):
52                    if after <= column:
53                        if g.node[(i, j)]['Color'] != g.node[(i,after)]['Color']:
54                            if g.node[(i,after)]['Circle'] == "C":
55                                g.add_edge((i,j),(i,after,"R"))
56                            else:
57                                g.add_edge((i,j),(i,after))
58            elif g.node[(i, j)]['Direction'] == "NW":
59                beforeRow = i-1
60                beforeColumn = j-1
61                while beforeRow >= 1 and beforeColumn >= 1:
62                    if g.node[(i, j)]['Color'] != g.node[(beforeRow,beforeColumn)]['Color']:
63                        if g.node[(beforeRow,beforeColumn)]['Circle'] == "C":
64                            g.add_edge((i, j), (beforeRow,beforeColumn, "R"))
65                        else:
66                            g.add_edge((i, j), (beforeRow,beforeColumn))
67                    beforeRow-=1
68                    beforeColumn-=1
69            elif g.node[(i, j)]['Direction'] == "SE":
70                afterRow = i+1
71                afterColumn= j+1
72                while afterRow <= row and afterColumn <= column:

```

```

13         if g.node[(i, j)]['Color'] != g.node[(afterRow, afterColumn)]['Color']:
14             if g.node[(afterRow,afterColumn)]['Circle'] == "C":
15                 g.add_edge((i, j), (afterRow, afterColumn, "R"))
16             else:
17                 g.add_edge((i, j), (afterRow, afterColumn))
18         afterRow+=1
19         afterColumn+=1
20     elif g.node[(i, j)]['Direction'] == "SW":
21         beforeRow = i+1
22         beforeColumn = j-1
23         while beforeRow <= row and beforeColumn >= 1:
24             if g.node[(i, j)]['Color'] != g.node[(beforeRow,beforeColumn)]['Color']:
25                 if g.node[(beforeRow, beforeColumn)]['Circle'] == "C":
26                     g.add_edge((i, j), (beforeRow, beforeColumn, "R"))
27                 else:
28                     g.add_edge((i, j), (beforeRow, beforeColumn))
29             beforeRow+=1
30             beforeColumn-=1
31     elif g.node[(i, j)]['Direction'] == "NE":
32         afterRow = i-1
33         afterColumn = j+1
34         while afterRow >= 1 and afterColumn <= column:
35             if g.node[(i, j)]['Color'] != g.node[(afterRow, afterColumn)]['Color']:
36                 if g.node[(afterRow, afterColumn)]['Circle'] == "C":
37                     g.add_edge((i, j), (afterRow, afterColumn, "R"))
38                 else:
39                     g.add_edge((i, j), (afterRow, afterColumn))
40             afterRow-=1
41             afterColumn+=1
42
43     #going through all the reverse nodes
44     if g.node[(i,j,"R")]['Direction'] == "N":
45         for before in range(1,i):
46             if g.node[(i,j,"R")]['Color'] != g.node[(before,j,"R")]['Color']:
47                 if g.node[(before, j,"R")]['Circle'] == "C":
48                     g.add_edge((i, j,"R"), (before, j))
49                 else:
50                     g.add_edge((i, j,"R"), (before, j,"R"), color='b')
51     elif g.node[(i, j,"R")]['Direction'] == "S":
52         for after in range(i+1,row+1):
53             if after <= row:
54                 if g.node[(i,j,"R")]['Color'] != g.node[(after,j,"R")]['Color']:
55                     if g.node[(after, j,"R")]['Circle'] == "C":
56                         g.add_edge((i, j,"R"), (after, j))
57                     else:
58                         g.add_edge((i, j,"R"), (after, j,"R"))
59
60     elif g.node[(i, j,"R")]['Direction'] == "W":
61         for before in range(1,j):
62             if g.node[(i, j,"R")]['Color'] != g.node[(i, before,"R")]['Color']:
63                 if g.node[(i,before,"R")]['Circle'] == "C":
64                     g.add_edge((i,j,"R"), (i,before))
65                 else:
66                     g.add_edge((i,j,"R"), (i,before,"R"))
67     elif g.node[(i, j,"R")]['Direction'] == "E":
68         for after in range(j+1,column+1):
69             if after <= column:
70                 if g.node[(i, j,"R")]['Color'] != g.node[(i,after,"R")]['Color']:
71                     if g.node[(i,after,"R")]['Circle'] == "C":
72                         g.add_edge((i,j,"R"), (i,after))
73                     else:
74                         g.add_edge((i,j,"R"), (i,after,"R"))
75     elif g.node[(i, j,"R")]['Direction'] == "NW":
76         beforeRow = i-1
77         beforeColumn = j-1
78         while beforeRow >= 1 and beforeColumn >= 1:
79             if g.node[(i, j,"R")]['Color'] != g.node[(beforeRow,beforeColumn,"R")]['Color']:
80                 if g.node[(beforeRow,beforeColumn,"R")]['Circle'] == "C":
81                     g.add_edge((i, j,"R"), (beforeRow,beforeColumn))
82                 else:
83                     g.add_edge((i, j,"R"), (beforeRow,beforeColumn,"R"))
84             beforeRow-=1
85             beforeColumn-=1
86     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

```