input\_text = input()  
  
# DO\_NOT\_EDIT\_ANYTHING\_ABOVE\_THIS\_LINE  
row = {  
 # "0": " \* "  
 # "1": " \*\*\* "  
 # "2": " \* \*\*\*\*\* "  
 # "3": " \*\*\* \*\*\*\*\*\*\* "  
 # "4": "\*\*\*\*\* \*\*\*\*\*\*\*\*\* "  
  
  
}  
  
  
# this is for drawing triangle of x height  
def draw\_triangle(height, max\_height, old\_height):  
 difference\_between\_them = max\_height-height  
 if difference\_between\_them > 0:  
 for empty\_place in range(difference\_between\_them):  
 if not empty\_place+old\_height in row.keys():  
 row[empty\_place+old\_height] = ""  
 for x in range(2\*height):  
 row[empty\_place+old\_height] = row[empty\_place+old\_height] + " "  
  
 for b in range(height):  
 new\_row = b + difference\_between\_them  
 if not new\_row+old\_height in row.keys():  
 row[new\_row+old\_height] = ""  
 for a in range(1, height - b):  
 row[new\_row+old\_height] = row[new\_row+old\_height] + " "  
 for c in range(2\*b + 1):  
 row[new\_row+old\_height] = row[new\_row+old\_height] + "\*"  
 for g in range(height-b):  
 row[new\_row+old\_height] = row[new\_row+old\_height]+" "  
  
  
def draw\_square(height\_square, max\_height, old\_height): # this is for drawing square  
 difference = max\_height - height\_square  
 if difference > 0:  
 for a in range(difference):  
 if not a+old\_height in row.keys():  
 row[a+old\_height] = ""  
 for c in range(height\_square+1):  
 row[a+old\_height] = row[a+old\_height] + " "  
 for x in range(height\_square):  
 new = x + difference  
 if not new+old\_height in row.keys():  
 row[new+old\_height] = ""  
 for y in range(height\_square):  
 row[new+old\_height] = row[new+old\_height] + "\*"  
 for t in range(1):  
 row[new+old\_height] = row[new+old\_height] + " "  
  
  
# this is for drawing rectangle of specific edges  
def draw\_rectangle(height\_rectangle, width\_rectangle, max\_height, old\_height):  
 difference = max\_height-height\_rectangle  
 if difference > 0:  
 for a in range(difference):  
 if not a+old\_height in row.keys():  
 row[a + old\_height] = ""  
 for f in range(width\_rectangle+1):  
 row[a + old\_height] = row[a + old\_height] + " "  
 for k in range(height\_rectangle):  
 new = k + difference  
 if not new+old\_height in row.keys():  
 row[new + old\_height] = ""  
 for i in range(width\_rectangle):  
 row[new + old\_height] = row[new + old\_height] + "\*"  
 for t in range(1):  
 row[new + old\_height] = row[new + old\_height] + " "  
  
  
# this is for empty rectangular  
def draw\_empty\_rectangular\_area(height\_rectangle\_e, width\_rectangle\_e, max\_height, old\_height):  
 difference = max\_height-height\_rectangle\_e  
 if difference > 0:  
 for a in range(difference):  
 if not a+old\_height in row.keys():  
 row[a + old\_height] = ""  
 for f in range(width\_rectangle\_e+1):  
 row[a + old\_height] = row[a + old\_height] + " "  
 for k in range(height\_rectangle\_e):  
 new = k + difference  
 if not new+old\_height in row.keys():  
 row[new + old\_height] = ""  
 for i in range(width\_rectangle\_e):  
 row[new + old\_height] = row[new + old\_height] + " "  
 for t in range(1):  
 row[new + old\_height] = row[new + old\_height] + " "  
  
  
def draw\_inverted\_triangle(width, max\_height, old\_height):  
 difference = max\_height - (width//2 + 1)  
 for e in range(width // 2 + 1):  
 new = e  
 if not new+old\_height in row.keys():  
 row[new + old\_height] = ""  
 for t in range(e):  
 row[new + old\_height] = row[new + old\_height] + " "  
 for y in range(width - (2 \* e)):  
 row[new + old\_height] = row[new + old\_height] + "\*"  
 for k in range(e + 1):  
 row[new + old\_height] = row[new + old\_height] + " "  
 if difference > 0:  
 for empty in range(difference):  
 if not (width//2 + 1)+empty + old\_height in row.keys():  
 row[(width//2 + 1)+empty + old\_height] = ""  
 for x in range(2\*(width//2 + 1)):  
 row[(width//2 + 1)+empty +  
 old\_height] = row[(width//2 + 1)+empty + old\_height] + " "  
  
  
def blank\_line(blank, max\_height, old\_height):  
 difference = max\_height  
 if difference > 0:  
 for a in range(difference):  
 if not a+old\_height in row.keys():  
 row[a + old\_height] = ""  
 for f in range(blank):  
 row[a + old\_height] = row[a + old\_height] + " "  
  
  
def add\_extra\_space(width\_of\_o, max\_height, old\_height):  
 difference = max\_height  
 if difference > 0:  
 for a in range(difference):  
 if not a+old\_height in row.keys():  
 row[a + old\_height] = ""  
 for u in range(width\_of\_o):  
 row[a + old\_height] = row[a + old\_height] + " "  
 for t in range(1):  
 row[a + old\_height] = row[a + old\_height] + " "  
  
  
def draw\_shapes(drawing, max\_height, old\_height):  
  
 if drawing.startswith("T"):  
 for\_reading\_second\_number = drawing.replace("T", "")  
 height\_triangle = int(for\_reading\_second\_number)  
 draw\_triangle(height\_triangle, max\_height, old\_height)  
  
 elif drawing.startswith("S"):  
 for\_reading\_second\_number\_square = drawing.replace("S", "")  
 height\_square = int(for\_reading\_second\_number\_square)  
 draw\_square(height\_square, max\_height, old\_height)  
  
 elif drawing.startswith("R"):  
 for\_reading\_second\_number\_rectangle = drawing.replace("R", "")  
 cancelling\_cross = for\_reading\_second\_number\_rectangle.split("x")  
 height\_rectangle = int(cancelling\_cross[0])  
 width\_rectangle = int(cancelling\_cross[1])  
 draw\_rectangle(height\_rectangle, width\_rectangle,  
 max\_height, old\_height)  
  
 elif drawing.startswith("E"):  
 for\_reading\_empty\_rectangle = drawing.replace("E", "")  
 cancelling\_cross\_2 = for\_reading\_empty\_rectangle.split("x")  
 height\_empty\_rectangle = int(cancelling\_cross\_2[0])  
 width\_empty\_rectangle = int(cancelling\_cross\_2[1])  
 draw\_empty\_rectangular\_area(  
 height\_empty\_rectangle, width\_empty\_rectangle, max\_height, old\_height)  
  
 elif drawing.startswith("V"):  
 for\_reading\_second\_number\_inverted\_triangle = drawing.replace("V", "")  
 width\_inverted\_triagle = int(  
 for\_reading\_second\_number\_inverted\_triangle)  
 draw\_inverted\_triangle(width\_inverted\_triagle, max\_height, old\_height)  
  
 elif drawing.startswith("B"):  
 blank = drawing.replace("B", "1")  
 blank\_real = int(blank)  
 blank\_line(blank\_real, max\_height, old\_height)  
  
 elif drawing.startswith("O"):  
 extra\_space = drawing.replace("O", "")  
 extra = int(extra\_space)  
 add\_extra\_space(extra, max\_height, old\_height)  
  
  
def finding\_max\_height(drawing):  
 if drawing.startswith("T"):  
 for\_reading\_second\_number = drawing.replace("T", "")  
 height\_triangle = int(for\_reading\_second\_number)  
 return height\_triangle  
  
 elif drawing.startswith("S"):  
 for\_reading\_second\_number\_square = drawing.replace("S", "")  
 height\_square = int(for\_reading\_second\_number\_square)  
 return height\_square  
  
 elif drawing.startswith("R"):  
 for\_reading\_second\_number\_rectangle = drawing.replace("R", "")  
 cancelling\_cross = for\_reading\_second\_number\_rectangle.split("x")  
 height\_rectangle = int(cancelling\_cross[0])  
 return height\_rectangle  
  
 elif drawing.startswith("E"):  
 for\_reading\_second\_number\_rectangle = drawing.replace("E", "")  
 cancelling\_cross = for\_reading\_second\_number\_rectangle.split("x")  
 height\_empty\_rectangle = int(cancelling\_cross[0])  
 return height\_empty\_rectangle  
  
 elif drawing.startswith("V"):  
 for\_reading\_second\_number\_inverted\_triangle = drawing.replace("V", "")  
 width\_inverted\_triagle = int(  
 int(for\_reading\_second\_number\_inverted\_triangle)//2 + 1)  
 return width\_inverted\_triagle  
  
 elif drawing.startswith("B"):  
 blank = drawing.replace("B", "1")  
 blank\_real = int(blank)  
 return blank\_real  
 elif drawing.startswith("O"):  
 extra\_space = drawing.replace("O", "")  
 extra = int(extra\_space)  
 return extra  
  
 elif drawing.startswith("DL"):  
 dash = drawing.replace("DL", "1")  
 dash\_line = int(dash)  
 return dash\_line  
  
  
def finding\_total\_width(drawing):  
 if drawing.startswith("T"):  
 for\_reading\_second\_number = drawing.replace("T", "")  
 height\_triangle = int(for\_reading\_second\_number)  
 width\_triangle = 2\*height\_triangle  
 return width\_triangle  
 elif drawing.startswith("S"):  
 for\_reading\_second\_number\_square = drawing.replace("S", "")  
 height\_square = int(for\_reading\_second\_number\_square)  
 width\_square = height\_square+1  
 return width\_square  
  
 elif drawing.startswith("R"):  
 for\_reading\_second\_number\_rectangle = drawing.replace("R", "")  
 cancelling\_cross = for\_reading\_second\_number\_rectangle.split("x")  
 width\_rectangle = int(cancelling\_cross[1])+1  
 return width\_rectangle  
  
 elif drawing.startswith("E"):  
 for\_reading\_second\_number\_rectangle = drawing.replace("E", "")  
 cancelling\_cross = for\_reading\_second\_number\_rectangle.split("x")  
 width\_empty\_rectangle = int(cancelling\_cross[1])+1  
 return width\_empty\_rectangle  
  
 elif drawing.startswith("V"):  
 for\_reading\_second\_number\_inverted\_triangle = drawing.replace("V", "")  
 width\_inverted\_triagle = int(  
 for\_reading\_second\_number\_inverted\_triangle)+1  
 return width\_inverted\_triagle  
  
 elif drawing.startswith("B"):  
 blank = drawing.replace("B", "1")  
 blank\_real = int(blank)  
 return blank\_real  
 elif drawing.startswith("O"):  
 extra\_space = drawing.replace("O", "")  
 extra = int(extra\_space)+1  
 return extra  
 elif drawing.startswith("DL"):  
 dash = drawing.replace("DL", "0")  
 dash\_line\_width = int(dash)  
 return dash\_line\_width  
  
  
max\_width = 0  
old\_height = 0  
shape\_row\_list = input\_text.split(",N,")  
for index in range(len(shape\_row\_list)):  
 current\_row\_shapes = shape\_row\_list[index]  
 current\_row\_shapes\_list = current\_row\_shapes.split(",")  
 max\_width\_current\_row = 0  
 for current\_shape in range(len(current\_row\_shapes\_list)):  
 item = current\_row\_shapes\_list[current\_shape]  
 item\_width = finding\_total\_width(item)  
 max\_width\_current\_row = max\_width\_current\_row + item\_width  
 if max\_width\_current\_row > max\_width:  
 max\_width = max\_width\_current\_row  
for index in range(len(shape\_row\_list)):  
 current\_row\_shapes = shape\_row\_list[index]  
 current\_row\_shapes\_list = current\_row\_shapes.split(",")  
 max\_height = 0  
 for current\_shape in range(len(current\_row\_shapes\_list)):#this is the part where I center my shapes  
 item = current\_row\_shapes\_list[current\_shape]  
 item\_height = finding\_max\_height(item)  
 if item\_height > max\_height:  
 max\_height = item\_height  
 if "DL" in current\_row\_shapes\_list:  
 dl\_count = current\_row\_shapes\_list.count("DL")  
  
 for index in range(dl\_count):  
 current\_row\_shapes\_list.remove("DL")  
  
 if not old\_height in row.keys():  
 row[old\_height] = ""  
 for t in range(max\_width-1):  
 row[old\_height] = row[old\_height] + "-"  
 old\_height = old\_height + 1  
 max\_width\_current\_row=0  
 for current\_shape in range(len(current\_row\_shapes\_list)):  
 item = current\_row\_shapes\_list[current\_shape]  
 item\_width = finding\_total\_width(item)  
 max\_width\_current\_row = max\_width\_current\_row + item\_width  
 shape\_order = int((max\_width - max\_width\_current\_row) / 2) +1  
 for index in range(old\_height,old\_height+max\_height):  
 if not index in row.keys():  
 row[index] = ""  
 for a in range(1,shape\_order):  
 row[index] = row[index] + " "  
 for current\_shape in range(len(current\_row\_shapes\_list)):  
 item = current\_row\_shapes\_list[current\_shape]  
 draw\_shapes(item, max\_height, old\_height)  
 old\_height = max\_height + old\_height  
for printing in range(len(row)):  
 print(row[printing])  
print("-" \* int(max\_width-1))  
  
  
  
# DO\_NOT\_EDIT\_ANYTHING\_BELOW\_THIS\_LINE