CODE:

```
import numpy as np
from itertools import repeat
from math import cos, sin, radians
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.patches import FancyArrowPatch
from mpl_toolkits.mplot3d import proj3d


def _plot(coords, transformed_coords):
   fig = plt.figure()
   ax = fig.add_subplot(111, projection='3d')
   ax.scatter3D(*zip(coords, transformed_coords))
   ax.plot(*zip(coords, transformed_coords), marker="<")
   ax.plot(*zip((0, 0, 0), transformed_coords))
   ax.plot(*zip((0, 0, 0), coords))


def init_transform(n=4):
     return np.identity(4)


def get_rotation_matrix(axis, t, in_degrees=True):
     # axis value can be 1, 2 or 3.
     axis = int(axis)
     # t is in degrees.
     if in_degrees:
     t = radians(t)

     rotation_matrices = {
     1: lambda t1: np.array([
               [1, 0,     0,         0],
               [0, cos(t1), -sin(t1), 0],
               [0, sin(t1), cos(t1),  0],
               [0, 0,     0,         1]
          ]),
     2: lambda t2: np.array([
               [cos(t2),  0, sin(t2), 0],
               [0,        1, 0,      0],
```

```
                [-sin(t2), 0, cos(t2), 0],
                [0,        0, 0,      1]
            ]),
        3: lambda t3: np.array([
                [cos(t3), -sin(t3), 0, 0],
                [sin(t3), cos(t3),  0, 0],
                [0,        0,         1, 0],
                [0,        0,         0, 1]
            ])
        }
        return rotation_matrices[axis](t)


def get_translation_matrix(x, y, z):
        return np.array([
        [1, 0, 0, x],
        [0, 1, 0, y],
        [0, 0, 1, z],
        [0, 0, 0, 1]
        ])


def get_coords():
        # returns coordinates in homogeneous reference.
        return [float(i) for i in input("Enter the
coordinates:").split()] + [1]


def algo(n_q, queries, coords):
        # taking the input in the format:
        # Q val: performing Q with val degrees. Q \in {R, T}

        TRANSFORMATIONS = {
        "R": get_rotation_matrix,
        "T": get_translation_matrix
        }

        T = init_transform()
        for query in queries:
        action = TRANSFORMATIONS.get(query[0])
        assert action is not None, "Invalid action given."
        T = np.matmul(T, action(*map(float, query[1:])))
```

```
        return np.matmul(T, coords)[:3]

if __name__ == '__main__':
        n_q = int(input("Enter the number of transformations: "))
        queries = [input().split() for _ in repeat(None, n_q)]
        coords = get_coords()
        transformed_coords = algo(n_q, queries, coords)
        print(transformed_coords)
        _plot(coords, transformed_coords)
```

Output:

```
Enter the number of transformations: 3
R 1 60
T 2 5 1
R 2 30
Enter the coordinates:4 1 2
Transformed coords: [6.46410162 2.3660254  5.56217783]
```