TASK 5

Issue 1: We have to account for cases where an atom have 2 outer connections

--- Step C: Process remaining unmapped nodes with a foreign connection to a non–ring (section 0) group of size 1

this is the current step C, we will have to change this,

new step C: process remaining unmapped nodes with 2 OR 1 foreign connection of non-ring of size 1

1. Now we will work with any other atoms with a foreign neighbor with size 1 and is non-ring that has not been mapped :

We will trace through all nodes in the section and change the following details

we need to add an intermediate step to check the atom[3] to see if it has 1 or 2 tuples, if it has 1 tuples:

we will proceed just like how we are doing right now. but when we check the inner neighbor foreign nodes, we also have to check neighbor[3] to see if it has 1 or 2 tuples, if it has 1 tuples we will proceed just like how we are doing right now, but if the amount of tuples is 2, we will consider putting it in the array if none of the tuple lead to a foreign section of size 1 that has not been mapped, with all of the other constraints still in place

note that the foreign section of size 1 that has not been mapped should be applied for all of the checking of foreign section of size 1

now we are done with that part, let's consider the case where atom[3] has 2 tuples, here we have 3 cases:

if no tuples lead to a non-ring section of length 1 that has not been mapped or only 1 tuples lead to a non-ring section of length 1 that has not been mapped, we proceed just like the previous steps

if both tuples lead to a non-ring section of length 1 that has not been mapped:

we check if all 3 atoms (the atom and 2 foreign atoms) are C, if they are assign them all as SC3 otherwise throw an error saying (2 size 1 neighbors but are not C and C)

issue 2: we need to finish the rest of the code for all other cases, namely:

elif count in (5, 4, 3, 2, 1):

raise ValueError("Non-benzene 6-ring mapping for remaining count {} is not yet implemented (TODO).".format(count))

this section

ok so let's start:make sure all atoms are either C or O otherwise return an error saying non-benzene 6 ring section has non C/O atom

if count = 5:

find an atom that is only connecting to 1 other atom that has not been mapped and is C: if can't find then throw an error

check the final[the neighbor that is already mapped] find all index in final that has that exact string, change the string first character to S (if not already), and assign it to all the indices as well as this atom.

then do the rest as it count = 4

if count = 4

trace through the section and for each atom

if it is connected to only 1 other atom that had not been mapped, check the atom type of both make them form a string (C and C would be CC)

check the dictionary for keys that start with T that has value[0] is 3 or 5, then check the value[2] and assign the key if the value[2] match with the string, forward or backward is ok

if count = 3

trace through the section and find the atom that is connected to 2 other atoms that had not been mapped

create a string that had the center atom type in the middle and the other atom types on each sides, ex. center atom is C and the two atoms are C and O then the string would be CCO, check the dictionary for keys that start with S that has value[0] is 3 or 5, then check the value[2] and assign the key if the value[2] match with the string, forward or backward is ok

if count = 2

    check the atom type of both make them form a string (C and C would be CC)

    check the dictionary for keys that start with T that has value[0] is 3 or 5, then check the value[2] and assign the key if the value[2] match with the string, forward or backward is ok

if count = 1

if it is O throw an error saying non benzene 6-ring has a lone Oxygen left

if it is C check the final[its neighbors] if any final start with a T, find all index in final that has that exact string, change the string first character to S, and assign it to all the index as well as this lone atom

if can’t find any then find any final start with a S and assign the whole final to this lone atom

issue 3: elif len(array1) == 2:

    raise ValueError("Non–ring section mapping too complex (foreign/inner connection, TODO).")

let’s do this

for this case, skip it and come back to after step C is finished and run step C again to see if len(array1) == 2, if it is, throw an error Non–ring section mapping too complex

TASK 6: fix the issue of ring precedence

issue: sometimes processing the right ring first will bring back a better result

fix: try each ring if it does not work move on

```
# Process benzene ring sections.
for i, section in enumerate(mapping):
    if section and section[0][2] == 2:
        if any(final[atom[0]] == "" for atom in section):
            final = map_benzene_ring_section(section, final,
martini_dict, mapping)
```

```
# Process non-benzene ring sections.
for i, section in enumerate(mapping):
    if section and section[0][2] == 1:
        if any(final[atom[0]] == "" for atom in section):
            size = len(section)
            if size == 6:
                final = map_nonbenzene_6_ring_section(section, final,
martini_dict, mapping)
            elif size == 5:
                final = map_nonbenzene_5_ring_section(section, final,
martini_dict, mapping)
            elif size == 3:
                final = map_nonbenzene_3_ring_section(section, final,
martini_dict)
            elif size == 4:
                final = map_nonbenzene_4_ring_section(section, final,
martini_dict)
            else:
                raise ValueError("Unexpected non-benzene ring section
size: " + str(len(section)))
```

so for these sections, we will do 2 takes, we will only "try" for the first take and if it works we will just do it if it lead to conflicts we skip that ring

then for the 2nd take if it still lead to conflict then we throw an error

next fix:

```
elif count == 3:
    # center atom has two unmapped neighbors
    center = next(atom for atom in remaining
                  if len([t for t in atom[4] if
final[ section[t[0]][0] ] == ""]) == 2)
    nbrs = [section[t[0]] for t in center[4] if
final[ section[t[0]][0] ] == ""]
    s = nbrs[0][1].upper() + center[1].upper() + nbrs[1][1].upper()
    for key,val in martini_dict.items():
        if key.startswith('S') and val[0] in (3,5) and (val[2] == s or
val[2] == s[::-1]):
            bead = key + generate_random_string()
            final[center[0]] = bead
            for n in nbrs: final[n[0]] = bead
            break
```

here we will have to check if it is already done mapped, if it is cool, if it is not we will:

check for if any non-center atom is a C, if it is C check the final[its neighbors] if any final start with a T, find all index in final that has that exact string, change the string first character to S, and assign it to all the index as well as this lone atom

if can't find any then find any final start with a S and assign the whole final to this lone atom

if no non-center atom is a C throw an error saying merging is not possible and mapping is also not possible

Task 6: Let's fix some quality of life changes:

```
               if foreign_atom[1].upper() == 'O':
                if bond_order == 2:
                    bead = "SN6a" + rstr
                elif bond_order == 1:
                    bead = "SN6" + rstr
                else:
                    bead = ""
            elif foreign_atom[1].upper() == 'N':
                bead = "SN6d" + rstr
            elif foreign_atom[1].upper() == 'S':
                bead = "SC6" + rstr
            elif foreign_atom[1].upper() == 'CL':
                bead = "SX3" + rstr
            elif foreign_atom[1].upper() == 'I':
                bead = "X1" + rstr
            elif foreign_atom[1].upper() == 'C':
                bead = "SC4" + rstr
            elif foreign_atom[1].upper() == 'BR':
                bead = "SX2" + rstr
            else:
                bead = ""
            final[atom[0]] = bead
            final[candidate] = bead
            final[foreign_atom[0]] = bead
```

There are too many of these sections across all section, lets create a function for them:

so there are 2 cases, the T beads and the S beads, we know whether it is T or S based on we already manually set all of them

let this function read the dictionary and

```
 martini_dict["SN6"] = [2, 0, "CC(O)", 2, ""]
 martini_dict["SN6d"] = [2, 0, "CC(N)", 2, ""]
 martini_dict["SC6"] = [2, 0, "CC(S)", 2, ""]
 martini_dict["SX3"] = [2, 0, "CC(Cl)", 2, ""]
 martini_dict["SX4e"] = [2, 0, "CC(F)", 2, ""]
 martini_dict["X1"] = [2, 0, "CC(I)", 2, ""]
 martini_dict["SX2"] = [2, 0, "CC(Br)", 2, ""]
 martini_dict["SC4"] = [2, 0, "CC(C)", 2, ""]
 martini_dict["SN6a"] = [2, 0, "CC(=O)", 2, ""]
 martini_dict["TN6"] = [2, 0, "C(O)", 2, ""]
 martini_dict["TX3"] = [2, 0, "C(Cl)", 2, ""]
 martini_dict["TN6d+"] = [2, 0, "C(N)", 2, ""]
```

```
    martini_dict["TC6+"] = [2, 0, "C(S)", 2, ""]

    martini_dict["TX1"] = [2, 0, "C(I)", 2, ""]

    martini_dict["TX2"] = [2, 0, "C(Br)", 2, ""]

    martini_dict["TC4"] = [2, 0, "C(C)", 2, ""]

    martini_dict["TN6a+"] = [2, 0, "C(=O)", 2, ""]

    martini_dict["TX4e"] = [2, 0, "C(F)", 2, ""]

    martini_dict["SN2a"] = [2, 0, "C(OC)", 2, ""]
```

look for the keys with first letter = T/S accordingly with value[0] = 2

it also takes in the input that is a string that is `foreign_atom[1].upper()`

```
for the case of O add = before O if bond length = 2


check this string with what is in () in value[2] and assign the key to
the atoms if it matches
```