Pseudo Code for Single layer winding

1. Take three input from user. First variable name is number of slots and its notation is number_of_slots, and its type is integer, and second input is number of poles and its notation is number_of_poles, and its type is integer, third variable name is number of phases and its notation is number_of_phases, and its type is integer but we take the number_of_phases are three.
2. Calculate some internal parameters:
    i. Number of coils and its notation is number of coils and it is calculated as
        number_of_coils = number_of_slots/2
    ii. Coil span and its notation is coil_span and it is calculated as
        coil_span = integer part of (number_of_slots/number_of_poles)
    iii. Coils per pole it is denoted as coils_per_pole and it is calculated as number_of_slots/2.
    iv. Calculate gcd1 of number of slots and number of poles then calculate a parameter named as factor and it is calculated as number_of_slots/(3*gcd1).
    v. Coils per phase and it is denoted as coils_per_phase and it can be calculated as number_of_slots/(2*3).
    vi. Motor periodicity and its notation is motor_periodicity and it is calculated as gcd(number_of_slots, number_of_poles//2).
    vii. Number of spokes and it is denoted as number_of_slots/motor_periodicity.
3. For single layer winding existence
    a. number of poles should be multiple of 2
    b. number of slots should be multiple of three.
    c. factor, number of coils, coils per phase, number of spokes and motor periodicity should be integer.
4. If conditions for existence of single layer winding are not full filled then return to user that single layer winding for this given number of poles and number of slots combination is not feasible.
5. Take a new variable named as slot pitch in mechanical degrees and it is notation is slot_pitch_mech and it is a global variable and its data type is float. it can be calculated as slot_pitch_mech = 360/number_of_slots.
6. Take another variable named as slot pitch in electrical degrees and it is denoted as slot_pitch_elec. It is global variable and its data type is float. It is calculated as slot_pitch_elec = (number_of_poles/2)*slot_pitch_mech.
7. Take another variable named as coil span and it is denoted as coil_span. It is global variable and its data type is float. It is calculated as coil_span = [number_of_slots/number_of_poles]. Take only integer part of coil_span.
8. Now take another variable named as coil pitch in electrical degrees and it is denoted as coil_pitch_elec. It is global variable and its data type is float. And it is calculated as coil_pitch_elec = coil_span * slot_pitch_elec.
9. Now make a list named as coil_number of length (number_of_slots+2)//2. And fill it from 1 to length.

10. Now make a new list named with coil_angle_mech and fill it with n*coil_pitch_mech for n in range(len(coil_number)) using list comprehension.
11. Now make a new list named with coil_angle_elec and fill it with n*coil_pitch_elec for n in range(len(coil_number)) using list comprehension.
12. Now make list named as theta which represents the slot angles. It is same as coil_angle_elec.
13. Now make a list named as list1 for handling the periodicity and define a variable iter_ with initial value as 1.
14. **START FOR LOOP** for i in range of number_of_slots/number_of_spokes:
15. temp = []
16. **START FOR LOOP** for j in range of int(number_of_spokes):
17. Append temp with iter_.
18. Iter_ += 1
19. Append list1 with temp
20. To handle odd length of list1 make a new list named as arr.
21. **START FOR LOOP** for ele in list1:
22. arr.append(ele[:len(ele)//2])
23. arr.append(ele[len(ele)//2:)
24. Now assign arr to list1.
25. Now define two list named as slotin and slotout. slotin will contain all elements with even indices in list1 and slotout will contain all elements with odd indices in list1.
26. **START FOR LOOP** for i in range(number_of_coils):
27. theta[i] = ((theta[i]+180) %360)-180
28. Make a new list named as thetai and thetai = [x+360 if x<0 else x for x in theta].
29. Define a new list named as theta1 that contains the sorted elements of thetai.
30. Make two new list named as slotin1 and slotout1.
31. set1= [False] * int(number_of_coils)
32. **START FOR LOOP** for i in range(len(theta1)):
33. for j in range(int(number_of_coils)):
34. if(len(slotin1)== int(number_of_coils)//3):
35. break
36. else:
37. if thetai[j]== theta1[i]:
38. if set1[j] ==False:
39. slotin1.append(slotin[j])
40.
41. slotout1.append(slotout[j])
42. set1[j]=True
43. Make two new list named as slotin2 and slotout2.
44. **START FOR LOOP** for i in range(len(theta1)):
45. for j in range(int(number_of_coils)):
46. if(len(slotin2)== int(number_of_coils)//3):
47. break
48. else:

```
            if thetai[j]== theta1[i]:
                if set1[j] ==False:
                    slotin2.append(slotin[j])
                    slotout2.append(slotout[j])
                    set1[j]=True
Make two new list named as slotin3 and slotout3.
START FOR LOOP for i in range(len(theta1)):
    for j in range(int(number_of_coils)):
        if(len(slotin3)== int(number_of_coils)//3):
            break
        else:
            if thetai[j]== theta1[i]:
                if set1[j] ==False:
                    slotin3.append(slotin[j])

                    slotout3.append(slotout[j])
                    set1[j]=True
RETURN slotin1,slotout1,slotin2,slotout2,slotin3,slotin3.
```