Pseudo Code for double 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.
2. Calculate some internal parameters:
   i. Calculate slot pitch in mechanical degree. Its notation is slot_pitch_mech and it is a global variable and its type is float type.
   ii. Take a variable number of slots per pole per phase and notation is number_of_poles_per_pole_per_phase, and its type is float. It is calculated as number_of_slots divided by number_of_poles and further divided by number_of_phases.
3. If conditions for existence of double layer winding are not full filled then return to user that double layer winding for this given number of poles and number of slots combination is not feasible.
4. The conditions for existence of double layer windings are:

   i. The motor should have number of phases equals to three.
   ii. Number_of_slots must be multiple of three.
   iii. number_of_slots_per_pole_per_phase should be less than or equal to 2.
   iv. Each back EMF must be shifted by 120 degree in Electrical degrees from the other two phases.
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 mechanical 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 take another variable named as chording angle and it is denoted as chording_angle and its data type are float. It can be calculated as chording_angle = (180 – coil_pitch_elec)/2.
10. Now define three lists named as slotin, slotout, and theta having length equal to the number_of_slots.
11. Define a new variable coil offset. It is denoted as coil_offset. Its type is float.
12. **Start FOR LOOP** from i=0:

13. Calculate coil offset which can be calculated as, coil_offset= $\frac{2}{3} \times \frac{number\_of\_slots}{number\_of\_poles} \times (1 + 3 \times q)$
14. IF coil_offset is integer:
15. Break;
16. **Start FOR LOOP** from i=1to number_of_slots:
17. Calculate Relative angle, theta[i]= $(i-1) \times \frac{number\_of\_slots}{number\_of\_poles} \times 180$
18. **Start FOR LOOP** from i=1to number_of_slots:
19. slotin[i]=i
20. slotout[i] + coil_span
21. If slotout[i] > number_of_slots:
22. slotout[i] = slotout[i] – number_of_slots
23. **Start FOR LOOP** from i=1to number_of_slots:
24. If theta[i] > 90:
25. slotin[i] = temp
26. slotin[i] = slotout[i]
27. slotout[i] = temp
28. **Start FOR LOOP** from i=1to number_of_slots:
29. If theta[i] > 90:
30. theta[i] = theta[i]-180
31. **if** theta[i] < -90:
32. theta[i] = theta[i]+180
33. Now initialize two lists named as slotin1 and slotout1 of size number_of_slots/3.
34. Take a new variable count of data type integer and it is private type variable. take initial value of this variable as zero.
35. **Start FOR LOOP** from i=1to (number_of_slots/3)-1:
36. list1 = [False for range (0, number_of_slots)]
37. Initialize a variable named as curr and its type is integer and private, curr = 0
38. diff = abs(theta[i] – curr)
39. if count <= (number_of_slots)/3:
40. **Start For LOOP** from i=0 to (number_of_slots)/3:
41. slotin1[i].append(slotin[j])
42. slotout[i].append(slotout[j])
43. count +=1
44. return slotin1, slotout1
45. now make four lists named as slotin2, slotout2, slotin3, and slotout3 and all of these lists have length of (number_of_slots/3). The type of the elements of these all lists is integer type.
46. **Start FOR LOOP** from i=1to (number_of_slots/3)-1:
47. slotin2[i] = slotin1[i] + coil_span
48. slotout2[i] = slotout1[i] + coil_span
49. while slotin2[i] > number_of_slots:

```
            Slotin2[i] = slotin2[i] – number_of_slots
        while slotout2[i] > number_of_slots:
            Slotout2[i] = slotout2[i] – number_of_slots
        slotin3[i] = slotin2[i] + coil_span
        slotout3[i] = slotout3[i] + coil_span
        while slotin3[i] > number_of_slots:
            slotin3[i] = slotin3[i] – number_of_slots
        while slotout3[i] > number_of_slots:
            slotout3[i] = slotout3[i] – number_of_slots
```

Finally return slotin1, slotout1, slotin2, slotout2, slotin3, and sloutout3