Algorithm overview:

The algorithm to solve the battery problem revolves around moving our batteries into a known state regardless of initial state. Then executing operations on that state that guarantee a success.

After spin zero we peek adjacent batteries, if they are the same charge we set that charge to positive. If they are opposing charges we change the negative to positive.

We then spin again, this time we peek opposite batteries. Again if they are the same polarity we set both to positive. If they are different we set the negative to positive.

It should be noted at this stage regardless of initial state we will now have 3 Positive batteries and one Negative battery

We then spin a third time and peek opposite batteries again. This time if the batteries are the same we will set 1 positive to negative. (The only match we could find at this point would be two positives). If they are different we change the negative to positive.

If we didn’t have a success out of the previous flip and subsequent spin we now will peek adjacent, and regardless of charge we will flip both batteries and spin again.

After the fifth spin if we don’t have a victory we will peek opposite this time, and flip both regardless of charge. This will guarantee a success and conclude our algorithm

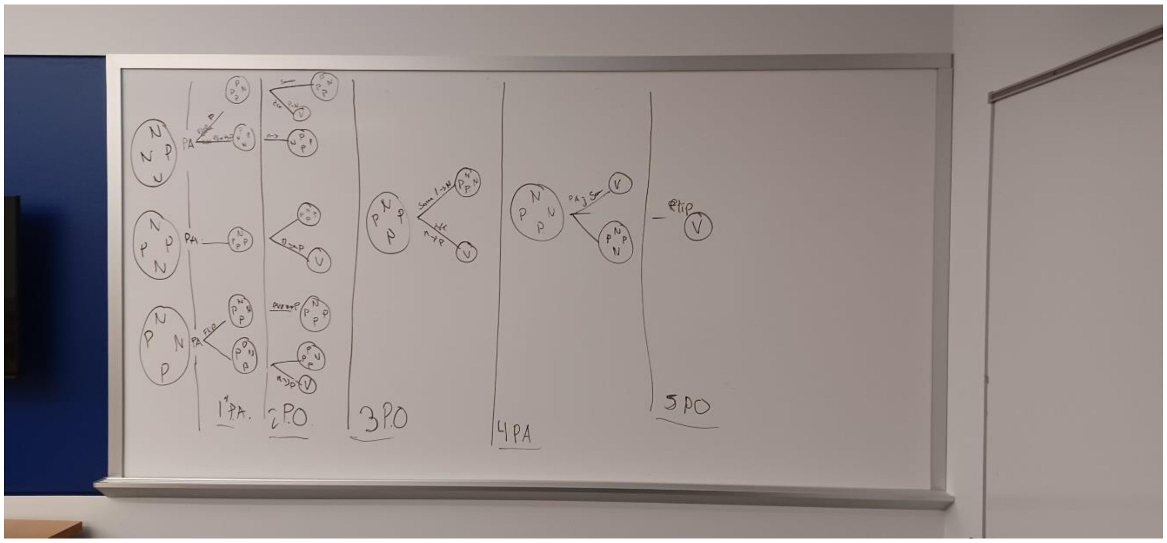
Pseudo-Code:

```
Initial device randomly = random( - - - -)

SPIN(device)  #spin will be defined program to spin the device for a random amount

If(device.powered()) #check for device success

        Print(“Got it”)
        exit()

PEEK(- - ?1 ?2)

CHANGE(- - P P)

SPIN(device) #spin 2

If(device.powered()) #check for device success

        Print(“Got it”)
        exit()


PEEK(? - ? -)
```

```
CHANGE (P - P - )

SPIN(device) #spin 3

If(device.powered()) #check for device success

    Print(“Got it”)
    exit()


PEEK(-  ?1 - ?2)

If (?1 == ?2):

    CHANGE(- P – N)

Else:

    CHANGE( - P - P)

SPIN(device)  #spin 4

If(device.powered()) #check for device success

    Print(“Got it”)
    exit()

PEEK(- - ?1 ?2)

CHANGE( - - FLIP, FLIP)

SPIN(device)  #spin 5

If(device.powered()) #check for device success

    Print(“Got it”)
    exit()

PEEK(?1 - ?2 -)

CHANGE( FLIP - FLIP - )

SPIN(device) #Final spin we will have guaranteed success after

If(device.powered()) #check for device success

    Print(“Got it”)
    exit()
```

Algorithm Correctness:

Ill be adding a figure to help explain. Essentially the algorithm in the first two steps either moves our initial 3 cases to a success state, or a state where we have some rotation of 3 Positives and one Negative. From here then knowing the sate of the device we can pick a path that guarantees a success. The next two moves either get a success or move us into a final known state. This final state we know the opposite batteries are set equal and we simply need to make them match the additional pair of opposite batteries This is done by and arbitrary flipping of the charges regardless of what charge we peek in this final fifth peek. Thus getting us a success at the most after five steps. The diagram below charts the path note circles with a 'V' in the center are success cases.



Algorithm Analysis:

This algorithm's Big-O is O(1) or constant time as we know we will achieve a success in at most five steps. The algorithm is deterministic when discussing a success for having all batteries in the same alignment. However, I believe if we needed a more constrained success of all the same charge and the charge specifically being P or N; this would be a non-deterministic algorithm and we would have to make a probabilistic heuristic argument.

Reflection:

This assignment I went directly to probabilistic problem solving. The notion of a nondeterministic element was quick to sway me into probability. It wasn't until we really looked at the solution space that a deterministic algorithm was even conceivable. Even after outlining the algorithm in class, I needed to walk through it myself to fully understand how it worked. Its interesting the way we control the flow getting some guaranteed knowledge and then operating in that space. The interesting way we converge to a single state regardless of initial state. I think in the future ill do more work to really consider the problem and solution space of a problem.