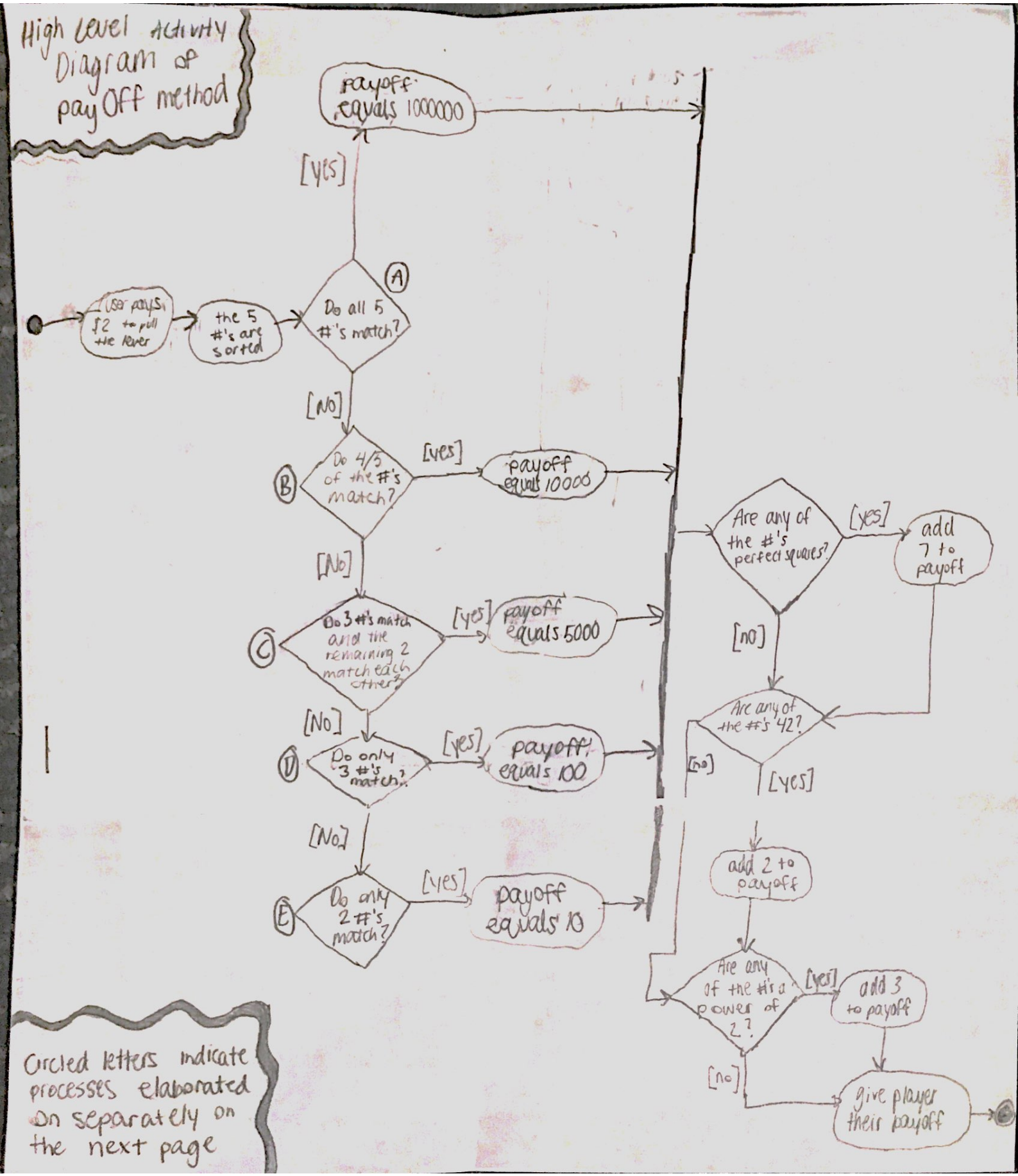

(A) Do all 5 #'s match?

- use a for loop to iterate through array and return false if array[0] != array[i]. Return true outside of this loop

(B) Do 4 out of 5 of the #'s match?

- use a [redacted] loop to iterate through the array and compare the second through fourth elements to the first element (since the array is sorted), and use an integer variable "count" to track the number of matches. The loop should terminate if array[0] ≠ array[i], or if "count" reaches 4. If the loop terminates and count ≠ 4, repeat the process by comparing the third through fifth elements to the second.

Ex: [1, 1, 1, 1, 2] would be for the initial process.

[1, 2, 2, 2, 2] would require the second process.

These are the only positions 4-matches could be in because the array is sorted. Return true or false when appropriate.

(C) Do 3 #'s match and the remaining 2 #'s match each other?

- use a similar process as (B), but terminate the initial loop if "count" = 3 or still if array[0] ≠ array[1]. If "count" ≤ 1, return false. If "count" = 2*, do the same process to check if the last 3 elements match by comparing the 4th and 5th elements to the 3rd, and track these matches with an integer variable "secondCount." This second iteration will terminate as soon as two values do not match, or when the end of the array is reached. Return true if "secondCount" = 3, otherwise return false.

* If "count" was equal to 3 (rather than 2), use these same general processes to check the condition and return true or false when appropriate.

Ex: [2, 2, 3, 3, 3] → case where "count" = 2 (and condition not)

[2, 2, 2, 3, 3] → case where "count" = 3 (and condition not)

(D) Do only 3 #'s match?

- Use a similar process as (C). This condition will just have more checks since there are more possible successful combinations of positions. If there are 3 matching numbers, they could be the first, middle, or last three elements of the array. True or false will be returned depending on the result.

(E) Do only 2 #'s match?

- use a nested for loop to compare the elements in the array and return true as soon as a match is found. If the loops terminate without a match, then return false.

The 6-8 conditions just require a simple for loop that will iterate through the array to check for the specified conditions. Each will stop checking as soon as the condition is met (if it actually is met) since the payoff amount applies only once per spin.

done for each element i in array until condition is met or all elements are processed

- 6 → `for (int j=1; i * i <= array[i]; j++){`
  `if (array[i] % j == 0 && array[i] / j == j)`
  `return true;`
  `}`
  `return false;`
- 7 → simple equality statement (`array[i] == 42`)
- 8 → keep dividing the element (`n = array[i]`) by two (`n = n/2`) while it `!= 1`. In any iteration, if `n % 2 != 0` and `n != 1`, then the element is not a power of 2. If n becomes 1 then it is a power of 2.