> No matter how one may think himself accomplished, when he sets out to learn a new language, science, or the bicycle, he has entered a new realm as truly as if he were a child newly born into the world. ~ Frances Willard, *How I Learned to Ride the Bicycle*

CS1010 Programming Methodology

Week 3: Computational Thinking/Algorithms (Selected Answers)

---

5. *N* white and black balls are arranged in a row. The example below shows a case of *N*=7.

Of course, we shall choose an appropriate notation to represent our objects (balls): **B** for black ball and **W** for white ball. So the above may be presented as BBWBWBW.

The task is to determine the least number of 'swaps' you need to shift all the white balls to the left of all the black balls, subject to the condition that you may only swap two neighbouring balls. For our example above, the least number of swaps you need is 9. (Work it out yourself! Try out other examples as well.)

Write an **algorithm** (not a C program!) to compute the answer. For this problem, a good algorithm needs not even carry out the swapping.

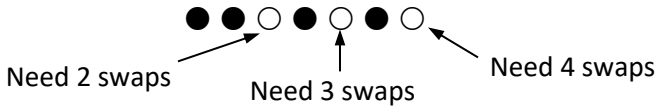
To help you start, here's a suggestion:

In solving algorithmic problem, we must name our data. Let's name the balls, from left to right, B_1, B_2, B_3, …, B_N for *N* balls.

Answer:

To be systematic, we shall examine the balls from left (starting from B_1) to right (ending at B_7 for our example).

Note that to shift the first white ball (B_3) to the left, we need 2 swaps because B_3 is at position 3 while its desired position is position 1, hence swaps needed = ball's position – desired position = 3 – 1 = 2. Similarly, the second white ball (B_5) requires 3 swaps as B_5 is at position 5 while its desired place is position 2, so swaps needed = 5 – 2 = 3. The third white ball (B_7) will need 7 – 3 or 4 swaps. Hence total number of swaps needed = 2 (for the first white ball) + 3 (for the second white ball) + 4 (for the third white ball) = 9.

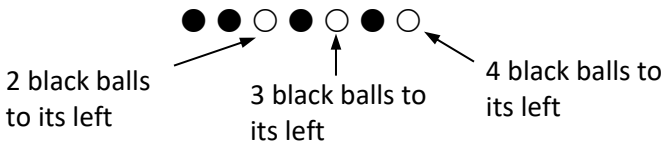


From the above discussion, it seems that the key lies in finding the difference between the position of a white ball and its desired position. How do we translate this idea (algorithm) into the pseudo-code?

Assuming that there are N balls B_1, B_2, … B_N.

```
countSwaps ← 0;
desiredPos ← 1;
for k from 1 to N
    if Bk is White
        swapsNeeded ← k – desiredPos;
        countSwaps ← countSwaps + swapsNeeded;
        desiredPos ← desiredPos + 1;
print countSwaps
```

Incidentally, we observe that the number of swaps needed for each white ball is also the number of black balls to its left. That is,



How do we write the pseudo-code that employs this idea?

```
countSwaps ← 0;
blackToLeft ← 0;
for k from 1 to N
    if Bk is Black
        blackToLeft ← blackToLeft + 1;
    else // Bk is White
        countSwaps ← countSwaps + blackToLeft;
print countSwaps
```

Are there other methods that scan the balls only once?