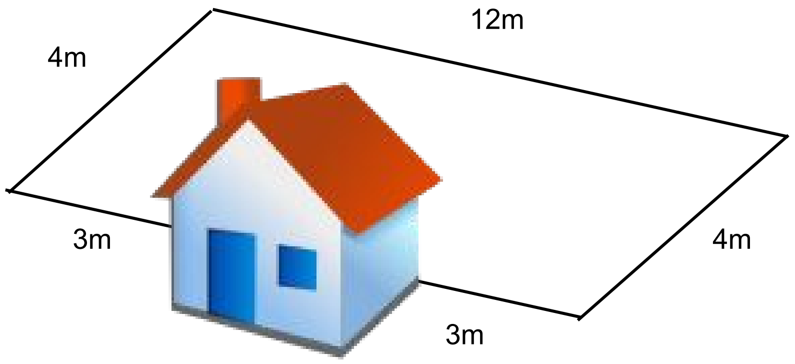
Problem of the Week
Problem A and Solution
The Fence

Problem

Agnes and her mother are building a fence around their backyard to keep their puppy safe. Their yard looks like the picture.

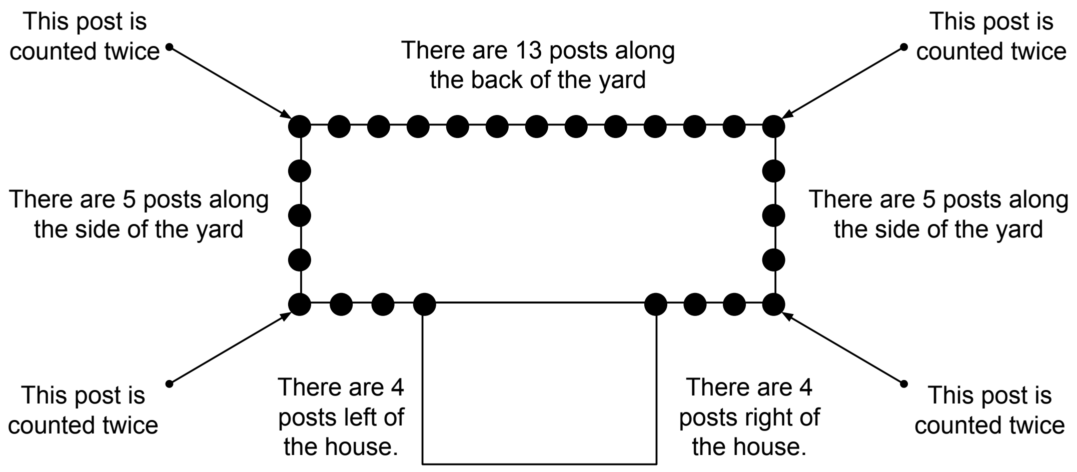


They start building the fence by placing one post at the back corner of the house. Then they add one fence post every meter around the yard until they put the final post at the other back corner of the house. Every corner of the yard has a post.

How many fence posts are needed altogether?

Solution

Here is a diagram of the layout for the fence posts.


This post is counted twice
There are 13 posts along the back of the yard
This post is counted twice
There are 5 posts along the side of the yard
There are 5 posts along the side of the yard
This post is counted twice
There are 4 posts left of the house.
There are 4 posts right of the house.
This post is counted twice

There are many ways to count the number of fence posts required. You can simply count the number of dots in the diagram to see that that Agnes and her mother need 27 posts.

Another way to calculate the total is to realize that since the fence posts are 1 m apart, it is easy to determine how many posts are required in each section of the backyard. They will need one more fence post than number of metres in each case. For example, for the 12 m across the back, they need 13 posts, since they need one at the first corner (0 metres), and one every metre after that until the second corner which is 12 metres away.

If they add these numbers up, the total is: $4 + 5 + 13 + 5 + 4 = 31$. However, the posts in each corner of the yard are counted twice using this method. Since 4 posts are counted twice, they need to subtract the 4 duplicates from the total calculated by adding the number of posts in each section. So the total number of posts required to build the fence is: $31 - 4 = 27$.

Teacher's Notes

This problem is a literal example of the fencepost problem, which is an analogy used to describe off-by-one errors in coding. These errors are very common, especially when dealing with a range of values where the beginning and end of the range are variables.

For example, in programming strings consist of a sequence of characters. We can think of each character as having a numbered position in the string. Normally, we start that numbering at 0, and the position is known as the index. So for example, the string ``Problem of the Week'' would be indexed as follows:

P	r	o	b	l	e	m		o	f		t	h	e		W	e	e	k
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Many programming languages include a function that extracts a substring by identifying the index positions of the characters you want from the original string. For example, you might have an expression like this:

```
substring(``Problem of the Week'', 15, 17)
```

Normally, the first number you provide to the substring function indicates index position of the first character you want and the second number is one more than the index position of the last character you want. So the substring in our example would be ``We''. This might seem odd, but this rule actually addresses the fencepost problem. Notice that the difference between the two numbers equals the number characters in the substring. So if we know the starting point of the substring, and the number of characters we want, then we simply add those two numbers together to find the second number for the substring function. Very often we know the starting position and the number of characters, so this can be the easier calculation.