Frankie Lloyd Wrighton, a little-known cousin of our chancellor, has conceived of a perfect tower whose face consists of stacked squares, where the side length of each square equals the length of the diagonal of the square above it. For example, if the top square has diagonal $a$, the second square will have side length $a$. Similarly, if the second square has diagonal $b$, the third square will have side length $b$, and so on. Let $A(s, n)$ be the total area of a perfect tower of $n$ squares, where the side length of the top square is $s$. In other words, $A(s,n)$ is the sum of the areas of the squares in the tower.

In its infinite wisdom, the University wants to build a perfect tower, but the board of trustees is concerned about the area of the face of the building. Write a correct Java method named `areaOfFrontOfBuilding` with the following specification. Use iteration. Do not use recursion and do not use `Math.pow` or any other exponentiation algorithm. Recall that `Math.sqrt(x)` returns the square root of $x$.

Parameters: $s$, the side length of the top square (a double) 
$n$, the number of squares in the tower (Assume $n > 0$. In the diagram, $n = 3$.)

Return: $A(s, n)$

```java
double areaOfFrontOfBuilding (double s, int n) {
    int numBlocks = 0;
    double side = s;
    double sum = 0;
    while (numBlocks < n) {
        sum = sum + side * side;
        numBlocks++;
        side = Math.sqrt(2) * side;
    }
    return sum;
}
```

Optional Extension:

a. State a sensible loop invariant for your implementation above.

1. $\text{sum} = A(s, k)$
2. $\text{side} = \text{length of side of the square numBlocks from the top}$

b. Use the loop termination condition and your loop invariant to argue that the method returns the correct value.

On termination, $k = n$

Substituting into 1 above, we have

$s = A(s, k) = A(s, n)$

Invariant 2 is useful to prove 1.