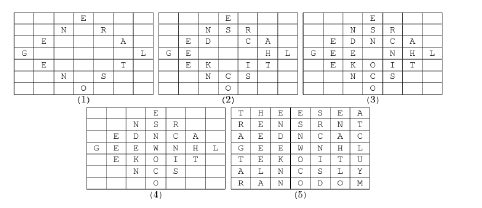
**Question:**

****ok now lets assume we have code that will append a fullstop to "GENERAL TSO NEEDS CHICKEN NOW" turning it into "GENERAL TSO NEEDS CHICKEN NOW.". now how do we determin the correct grid dimentionsion ? I assume that we must start with the middle of the leftmost column, traverse diagonally until the middle colu,n of the upermost row...etc until the last character (ie the dot) ends up in the center of the grid. is my understanding correct ?  
  
**Answer:  
📌 Your example**

* Input: "GENERAL TSO NEEDS CHICKEN NOW"
* Remove spaces: "GENERALTSONEEDSCHICKENNOW" (25 letters)
* Add full stop → "GENERALTSONEEDSCHICKENNOW." (26 letters total)

So your **message length** is **28**.

## ✅ ****How you determine the grid dimensions****

**You don’t pick grid dimensions to force the final character (the .) into the exact center.**

Instead, you:  
1️⃣ **Calculate the length** of the final cleaned message.  
2️⃣ Find the **smallest odd perfect square** that can hold it.

* This is required so that the grid is square and the diamond pattern works symmetrically.
* The pattern always starts at the **middle cell of the leftmost column**, not the center of the grid.

## 📏 ****Step-by-step****

**Example:**

* Length = 28
* Perfect odd squares:
  + 5×5 = 25 → too small
  + 7×7 = 49 → ✔️ big enough

So you use a **7×7 grid**.  
That’s the correct grid **dimension**.

## 🔑 ****How the diamond traversal works****

1. Start at the **middle cell of the leftmost column**:
   * For 7×7, the rows are indexed 0 to 6.
   * The middle row is row = 3.

So you start at grid[3][0].

1. From there, you follow the diamond pattern:
   * Up-right diagonally until you hit the top edge.
   * Then down-right diagonally until you hit the right edge.
   * Keep spiraling inward: new diamond, same idea.

✅ **The diamond shape does not force the final character to land in the center.**

* Where the . lands depends on how many cells the diamond path covers.
* If your message length is less than the total grid cells, you’ll have filler random letters for any leftover spots.

## 🔍 ****So is your understanding correct?****

**Almost!** You got the **start point and diamond shape right**, but:

* You **do not adjust the grid dimension** to make the last character hit the center.
* You just find the smallest odd square big enough.
* The diamond pattern naturally works inward — the last real character may or may not land in the center.

✅ In this example:

* 28 letters means you’ll fill about half the grid (49 total slots).
* The last real character (.) lands wherever the diamond traversal’s 28th step lands.
* Any remaining cells get random filler.

## 🗝️ ****Key takeaway****

✔️ The **grid size** is always the **smallest odd square** that fits the total length (including the .).  
✔️ The **diamond traversal** naturally decides where the . goes — you don’t manually align it to the center.  
✔️ The **center cell** often ends up being near the last real letter for short-ish messages — but that’s a side effect of the pattern, not a strict rule.

## ✅ ****So: your steps are:****

1. Clean input → remove spaces → add . if needed.
2. Compute length.
3. Pick smallest odd square ≥ length.
4. Create the grid.
5. Start diamond pattern at middle cell of leftmost column.
6. Insert letters in pattern order.
7. Fill leftover cells with random A–Z.
8. Output the encrypted string by reading column-by-column.

## ✔️ Your understanding is almost perfect!

Would you like a **tiny visual or code snippet** that calculates grid size + shows the first diamond coordinates for a small example?