**Stone Game III:**

Alice and Bob continue their games with piles of stones. There are several stones **arranged in a row**, and each stone has an associated value which is an integer given in the array stoneValue.

Alice and Bob take turns, with **Alice** starting first. On each player's turn, that player can take **1, 2 or 3 stones** from the **first** remaining stones in the row.

The score of each player is the sum of values of the stones taken. The score of each player is **0** initially.

The objective of the game is to end with the highest score, and the winner is the player with the highest score and there could be a tie. The game continues until all the stones have been taken.

Assume Alice and Bob **play optimally**.

Return *"Alice"* if Alice will win, *"Bob"* if Bob will win or *"Tie"* if they end the game with the same score.

**Example 1:**

**Input:** values = [1,2,3,7]

**Output:** "Bob"

**Explanation:** Alice will always lose. Her best move will be to take three piles and the score become 6. Now the score of Bob is 7 and Bob wins.

**Example 2:**

**Input:** values = [1,2,3,-9]

**Output:** "Alice"

**Explanation:** Alice must choose all the three piles at the first move to win and leave Bob with negative score.

If Alice chooses one pile her score will be 1 and the next move Bob's score becomes 5. The next move Alice will take the pile with value = -9 and lose.

If Alice chooses two piles her score will be 3 and the next move Bob's score becomes 3. The next move Alice will take the pile with value = -9 and also lose.

Remember that both play optimally so here Alice will choose the scenario that makes her win.

**Example 3:**

**Input:** values = [1,2,3,6]

**Output:** "Tie"

**Explanation:** Alice cannot win this game. She can end the game in a draw if she decided to choose all the first three piles, otherwise she will lose.

**Example 4:**

**Input:** values = [1,2,3,-1,-2,-3,7]

**Output:** "Alice"

**Example 5:**

**Input:** values = [-1,-2,-3]

**Output:** "Tie"

**Constraints:**

* 1 <= values.length <= 50000
* -1000 <= values[i] <= 1000