Problem Statement

Bolt and Thunder take turns playing a game, with Bolt starting first.

Initially, there are n stones in a pile. On each player's turn, that player makes a move consisting of removing any non-zero square number of stones in the pile.

Also, if a player cannot make a move, he/she loses the game.

Given a positive integer n, return true if and only if Bolt wins the game otherwise return false, assuming both players play optimally.

Input Format

First line contains single integer n.

Output Format

Print true if and only if Bolt wins the game otherwise return false

Constraints

- $1 <= n <= 10^5$

Sample Testcase 0

Testcase Input

4

Testcase Output

True

Explanation

n is already a perfect square, Alice can win with one move, removing 4 stones (4 → 0).

Sample Testcase 1

Testcase Input

1

Testcase Output

True

Explanation

Alice can remove 1 stone winning the game because Bob doesn't have any moves.

CODE-

```cpp
#include <iostream>
#include <vector>

using namespace std;

bool canWin(int n) {
    // Create a vector to store the results of subproblems
    vector<bool> dp(n + 1, false);

    // Base case: if there are 0 stones, Bolt loses
    dp[0] = false;

    // Calculate the results for all numbers from 1 to n
    for (int i = 1; i <= n; i++) {
        int j = 1;
        while (j * j <= i) {
            // If Bolt can find a move that leads to a losing position for Bob, he wins.
            if (!dp[i - j * j]) {
                dp[i] = true;
                break;
            }
            j++;
        }
    }

    // The final result is stored in dp[n]
    return dp[n];
}

int main() {
    int n;
    cin >> n;
    // Output
    cout << (canWin(n) ? "True" : "False") << endl;

    return 0;
}
```