Problem: The Word Search (Backtracking)

Difficulty: Medium

Time Limit: 1.0s

Memory Limit: 256MB

1. Input Format (for C++ `cin`)

1. **First Line:** Two integers `R` and `C` (Rows and Columns).
2. **Next `R` Lines:** The grid content. Each line contains `C` space-separated characters.
3. **Last Line:** A single string `Word` (the target).

Output: Print `1` if found, `0` if not found.

2. Test Cases

Test Case 1: The Standard Snake

Path: (0,0) -> (0,1) -> (0,2) -> (1,2) -> (2,2) -> (2,1)

Input:

```
3 4
A B C E
S F C S
A D E E
ABCCED
```

Output:

```
1
```

Test Case 2: The "Visited" Constraint

Logic: Path A->B->C is valid, but it cannot go back to B because B is already visited in current recursion stack.

Input:

```
3 4
A B C E
S F C S
A D E E
ABCB
```

Output:

```
0
```

Test Case 3: The Backtrack (Dead End)

Logic: The algorithm might try `S -> Right(E)`. It realizes that leads nowhere. It must backtrack to `S` and try `S -> Down(D)` (or similar path depending on grid).

Input:

```
3 4
A B C E
S F C S
A D E E
SEE
```

Output:

```
1
```

Test Case 4: Edge Case (Smallest Grid)

Logic: 1x1 grid where the letter matches.

Input:

```
1 1
A
A
```

Output:

```
1
```

3. C++ Parsing Logic Helper

If you are using `cin`, your loop should look like this:

```
int R, C;
cin >> R >> C;
vector<vector<char>> board(R, vector<char>(C));

for(int i = 0; i < R; i++) {
    for(int j = 0; j < C; j++) {
        cin >> board[i][j];
    }
}
```

```
        string word;
        cin >> word;
```