Jump Game

Matthew's Stats

- Time taken: 40 minutes
- Files: 3
- Lines of Code: 116 including whitespace and comments

What to Submit

- A zip file containing
  - Your .cpp and .h files that make up your solution
  - A CMakeLists.txt file that will generate an executable named **JumpGame** from your .cpp and .h files
    - In your CMakeLists.txt in the add_executable line, make sure you have `add_executable(`**`JumpGame`**` your_.cpp_files your_.h _files)` so the correctly named executable gets built
- Make sure to zip the files you want to submit and **NOT** the folder that contains the files. Submitting the folder with the files will cause your program to fail to build.

Restrictions and Requirements

- No global variables may be used
- Your submission must contain at least 2 or more .cpp files and one or more .h files

Problem Description

You are given a list of integer numbers. Each number tells you how far to the right or left you can move from that spot. You can only move exactly that many spaces. Moving from one number to the next is considered a move. Your goal is to find the sequence of moves that get you from the first number to the last number in the fewest moves possible. If however, there is no sequence of moves that will get you to the end, then you should state that. For example, if you were given the list

Index	0	1	2	3	4	5	6
Value	1	2	4	1	1	1	7

Then the smallest sequence of moves that gets you from index 0 to index 6 is {0, 1, 3, 2, 6}.

Problem Details

Input

Command Line Arguments

- Will always be valid
- The list of integers to solve
    - Each integer will have a value greater than or equal to 0

Standard Input

- None

Implementation

- You will find recursion to be very helpful in solving this problem.
- When solving the problem, first go to the left and then go right
    - This will help us to have the same solution if there are multiple equally valid solutions
- If a move would take you out of bounds of the list, before the first entry or after the last entry, it is invalid and should not be taken

Example

- Input has been underlined to help you differentiate between input and output.
    - You do not need to underline anything
- I've also provided an example executable named ExampleJumpGame that you can run by doing `./ExampleJumpGame numbers`
    - Numbers is the list of numbers to find the solution to
    - It is only guaranteed to run on Kodethon and may not run on your personal computer

Example 1

```
./JumpGame 1 2 4 1 1 1 7
The solution is: {0, 1, 3, 2, 6}
```

Example 2

```
./JumpGame
There is no solution to the given game.
```