Ex:

```
function vecout = compress(vec)

  vecout = vec;

  for index = 1:length(vec) - 1
    if vecout(index:index + 1) == [1, 0];
      vecout(index:index + 1) = [0, 1];
      vecout = vecout      % Prints out vecout.
    end
  end

  return
end
```

For the above Matlab® function, find the result of the following commands:

```
>> vec = [1, 0, 0, 1];
>> compress(vec)
```

Sol'n: This function scans `vec` for the pattern [1, 0] and pushes the 1 to the right by replacing [1, 0] with [0, 1]. In this problem, the function is only run once, but if called many times could push the 1's in a pattern to the right end of `vec` (without changing the total number of 1's and 0's).

The first time thru the loop, the [1, 0] pattern will be found and replaced with the [0, 1] pattern. This happens before the value of `vecout` is displayed.

```
vecout =
     0     1     0     1
```

The second time thru the loop, the [1, 0] pattern will again be found because of the change in `vecout` and the new position (2nd and 3rd bits) where `vecout` is being examined. The [1, 0] pattern is changed to [0, 1], shifting the 1 another position to the right.

```
vecout =
     0     0     1     1
```

The third and last time thru the loop, the [1, 0] pattern is not found at bits 3 and 4, so `vecout` is unchanged. When the function returns, it prints an answer:

```
ans =
     0     0     1     1
```