String compression

chars = ["a", "a", "b", "b", "c", "c", "c"]

compress
1. eg : ["a", "a"] ⇒ a2
2. eg : ["a"] ⇒ a

a2b2c3

↓

['a', '2', 'b', '2', 'c', '3'] → return length → 6

['a', 'b', 'b', 'a', 'a']

↓

ab2a2

↓

5 ∵ continuous lone chaiye

phele tho count all continuous chars

char = a a b b c c c (indices 0 1 2 3 4 5 6)

1, 2, X

replace a at 1st index with count

a a b b c c c (indices 0 1 2 3 4 5 6)

1, 2, 1, 2, X

replace b at 2nd index with count

....

a2 b2 c3

Psuedo code

```
idx = 0
for (i=0; i<n; i++) {
    ch = chars[i]
    count = 0;
    while (i<n && chars[i]==ch) {
        count++;
        i++;
    }
    if (count == 1) ⇒ chars[idx++] = ch;
    else {
        chars[idx++] = ch;
        string str = to_string(count)
        for (char dig : str)
            char[idx++] = dig;
    }
    i--;
}
```

Dry run

[a | a | b | b | c | c | c] (idx → 0, i → 0)

ch = a count = 1

[a | a | b | b | c | c | c] (idx → 0, i → 1)

ch = a count = 2

[a | a | b | b | c | c | c] (idx → 0, i → 2)

b ≠ a

∴ idx mein save 'a' then do idx++

[a | a | b | b | c | c | c] (idx → 1, i → 2)

convert count to str then store now do i-- (meaning it'll stay there only once it gets countered)

[a | 2 | b | b | c | c | c] (idx → 2, i → 2)

ch = b count = 1

[a | 2 | b | b | c | c | c] (idx → 2, i → 3)

ch = b count = 2

[a | 2 | b | b | c | c | c] (idx → 2, i → 4)

c ≠ b

at idx save 'b' then do idx++

[a | 2 | b | b | c | c | c] (idx → 3, i → 4)

convert count to str and save at idx then do idx++

i-- so i wont be i++ and it'll stay at current position even after loop over

[a | 2 | b | 2 | c | c | c] (idx → 4, i → 4)

ch = c count = 1

[a | 2 | b | 2 | c | c | c] (idx → 4, i → 5)

ch = c count = 2

[a | 2 | b | 2 | c | c | c] (idx → 4, i → 6)

ch = c count = 3

now loop over so save ch at idx do idx++, convert count to str and done

[a | 2 | b | 2 | c | 3 | c] (idx → 6, i → 6)

now resize vector then return idx value because that's basically the length lol