Exercise 1.3.45: Stack Generability

Determine whether intermixed sequence causes stack to underflow. Devise a linear-time algorithm that determines whether permutation can be generated as output by our client

Solution:

The stack does not underflow unless first k-pop operations occur before first first k-push operations as initially the stack is empty.

```
//Pseudocode soln - check underflow
function underflow(String[] inputs)
    int pushes;
    int pops;

    for i in inputs:
        if(i equals '-')
            pops++

        else if (i equals '+')
            pushes++

    if (pushes < pops)
        return true // stack underflows

    else
        return false //stack is stable
```

```
//check if permutation possible
stack = []
top = -1

for i in permutation:
    integer_val = (int) i
    if(stack.isEmpty() or integer_val > stack.peak())

        while(top < integer_val)
            top++
            stack.append(top) //keeps track of stack positions where pops can occur in accending order

        else if(integer_val equals stack[-1])
            stack.pop()

        else
            return false //position can't be popped as it comes before current position
                         // current top value must be popped first.

        return true //if elements in stack can be popped in correct order for permuation
```