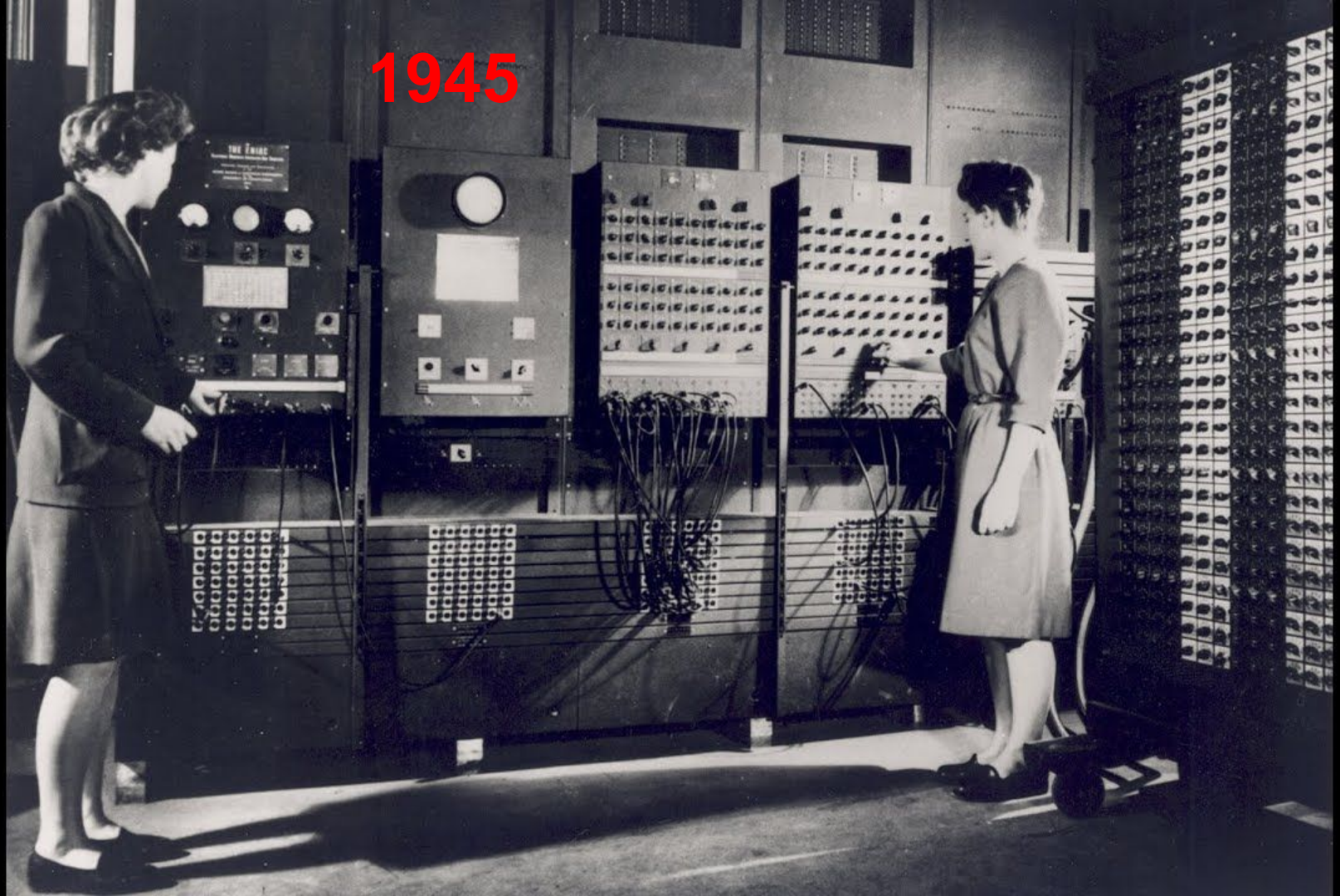
Generative AI And Programming

Peter Norvig

Stanford University
Human-Centered
Artificial Intelligence

Google

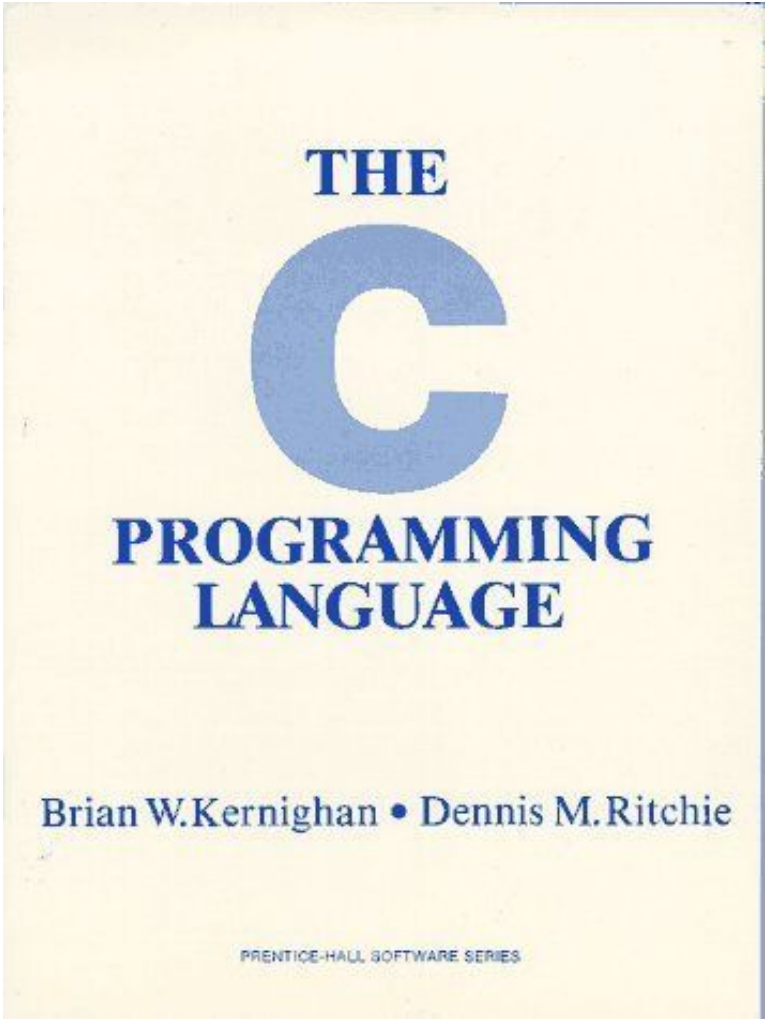
A Short History of Software Engineering


1945


AT&T
UNIX® System V

1978

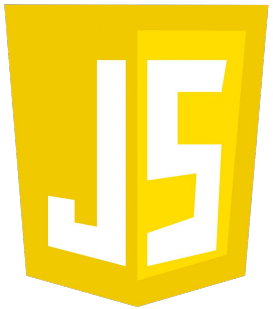

digital
pdp11


THE
C
PROGRAMMING
LANGUAGE
Brian W. Kernighan • Dennis M. Ritchie
PRENTICE-HALL SOFTWARE SERIES

2014

2023

JavaScript



GitHub Copilot

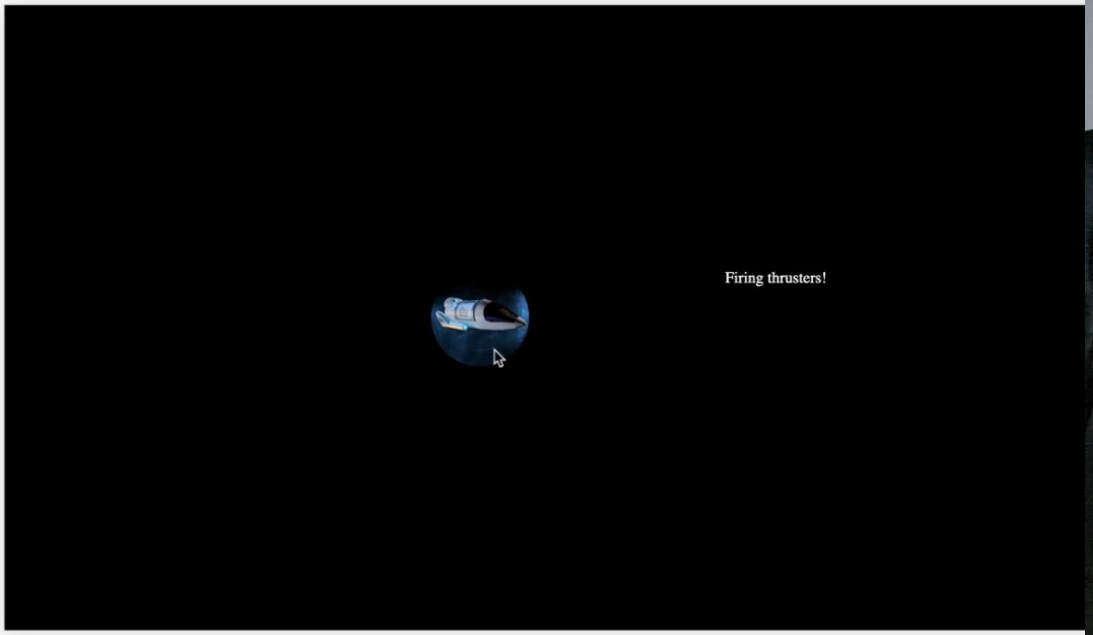
Will Coding Jobs Cease to Exist in Three Years?

ajitjaokar | April 25, 2023 at 11:07 am

1970s	2000s	2020s
Optimize hardware	Optimize programmer	Assist user
Low-level code	High-level code	Dialog: NL, code, diagrams, …
Simple problems	Complex problems	Wicked problems
Ad hoc	Mathematical/Logical	Natural science/empirical
On your own	Standard methodology	Executable methodology

Q: How is Automated Programming used today?

A: (1) API lookup (pair programming)
(2) Problem solving (solo programming)



When the rocket is clicked, temporarily display some text saying "Firing thrusters!" in white on the current location -- and temporarily speed up by 4x for 0.25 second.

D.Backspace

You are given two strings s and t, both consisting of lowercase English letters. You are going to type the string s character by character, from the first character to the last one.

When typing a character, instead of pressing the button corresponding to it, you can press the "Backspace" button. It deletes the last character you have typed among those that aren't deleted yet (or does nothing if there are no characters in the current string). For example, if s is "abcbd" and you press Backspace instead of typing the first and the fourth characters, you will get the string "bd" (the first press of Backspace deletes no character, and the second press deletes the character 'c'). Another example, if s is "abcaa" and you press Backspace instead of the last two letters, then the resulting text is "a".

Your task is to determine whether you can obtain the string t, if you type the string s and press "Backspace" instead of typing several (maybe zero) characters of s.

Input

The first line contains a single integer q ($1 \le q \le 10^5$) — the number of test cases.

The first line of each test case contains the string s ($1 \le |s| \le 10^5$). Each character of s is a lowercase English letter.

The second line of each test case contains the string t ($1 \le |t| \le 10^5$). Each character of t is a lowercase English letter.

It is guaranteed that the total number of characters in the strings over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print "YES" if you can obtain the string t by typing the string s and replacing some characters with presses of "Backspace" button, or "NO" if you cannot.

You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

Input

```
4
ababa
ba
ababa
bb
aaa
aaaa
aababa
ababa
```

Output

```
YES
NO
NO
YES
```

Note

Consider the example test from the statement.

In order to obtain "ba" from "ababa", you may press Backspace instead of typing the first and the fourth characters.

There's no way to obtain "bb" while typing "ababa".

There's no way to obtain "aaaa" while typing "aaa".

In order to obtain "ababa" while typing "aababa", you have to press Backspace instead of typing the first character, then type all the remaining characters.

Name "t" re-used for two different variables, an int and a str

Explain the code: comments and/or docstring

One-letter variable names


```
t=int(input())
for i in range(t):
    s=input()
    t=input()
    a=[]
    b=[]
    for j in s:
        a.append(j)
    for j in t:
        b.append(j)
    a.reverse()
    b.reverse()
    c=[]
    while len(b)!=0 and len(a)!=0:
        if a[0]==b[0]:
            c.append(b.pop(0))
            a.pop(0)
        elif a[0]!=b[0] and len(a)!=1:
            a.pop(0)
            a.pop(0)
        elif a[0]!=b[0] and len(a)==1:
            a.pop(0)
    if len(b)==0:
        print("YES")
    else:
        print("NO")
```

First AlphaCode reads the two phrases.

Python standard style is to surround operators with spaces, e.g. "a = []"

These 10 lines could be just:
a = list(reversed(input()))
b = list(reversed(input()))

Usually, j iterates over ints, not characters

Doing b.pop() is an O(1) operation. But b.pop(0) is O(n). Shouldn't have reversed lists!

"c" is set up as a stack to hold elements popped off of "b", but then "c" is never used elsewhere

If the letters at the end of both phrases don't match, the last letter must be deleted. If they do match we can move onto the second last letter and repeat.

Backspace deletes two letters. The letter you press backspace instead of, and the letter before it.

These 5 lines could be just:
"else:
    a = a[2:]"

Encapsulate as a function with signature backspacer(s: str, t: str) → bool

Straight-line top-level code is harder to test

If we've matched every letter, it's possible and we output that.

Better Answer

“Here is a function to solve the problem. You can see tests and timing here, or an informal argument for correctness here, or compare to a simpler but slower version, or a faster but more complex version.”

```
def backspacer(source: str, target: str) -> bool:
    """Can you obtain the string `target`, if you type the string `source` and
    press "Backspace" instead of typing several characters of `source`?"""
    while source and not source.endswith(target):
        if source[-1] == target[-1]:
            source, target = source[:-1], target[:-1] # Match end characters
        else:
            source = source[:-2]                      # Press backspace instead
    return source.endswith(target)
```

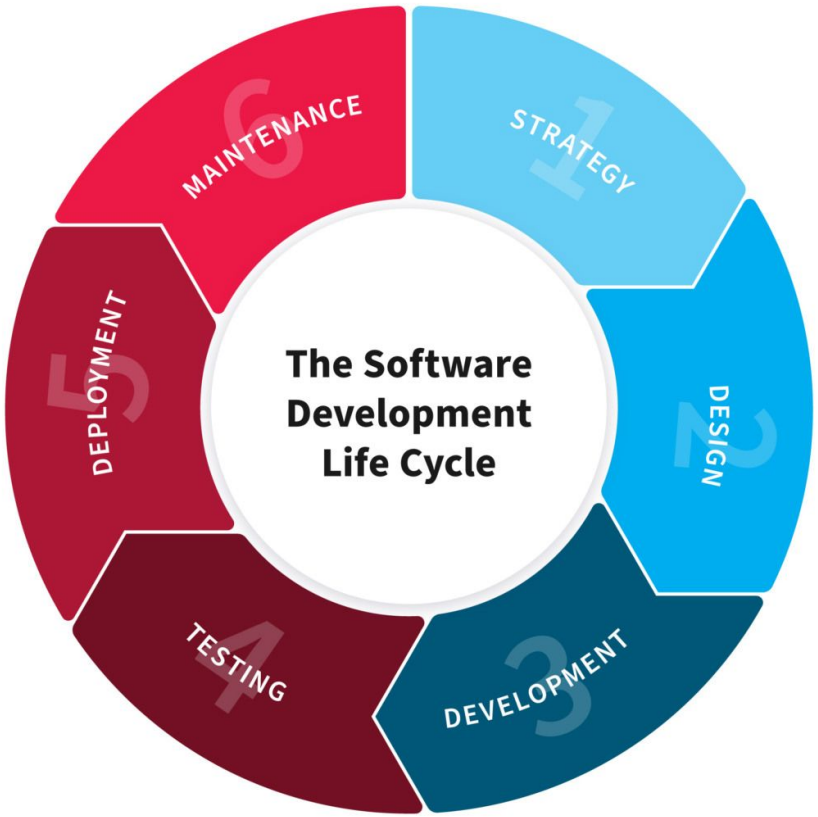
THE LAKE WOBEGON EFFECT

Half the programmers are below average

Q: How could Automated Programming be used tomorrow?

Automation throughout the SW Lifecycle

- **Strategy**: help define objectives
- **Design**: create and verify specifications
- **Development**: automatically write or autocomplete code
- **Testing**: automate tests
- **Deployment**: gather feedback, federated learning
- **Maintenance**: fast, correct updates, retraining instead of re-coding and re-releasing



Automation throughout the SW Lifecycle