Name: NOEL BOURKE

Date: 14/03/2016

Student No.: 112 493 412

20/20

David Pulay

Solution

I was uncertain about the instructions, so I have two programs - one which ignores punctuation and spaces and another which assumes they won't be input. These are called "With Checking" and "Without Checking" respectively.

Space:

The "With Checking" program takes up 67 bytes (including the end command and the 00 value placed on the stack by the first instruction) and so can fit a palindrome of 125 characters without overwriting the VDU or the program. Note this is 125 letters, i.e. not counting spaces + punctuation.

The "Without Checking" program takes up 44 bytes of memory (including the 'end' and 00 as above) and so can check a palindrome of up to 148 characters without error. ✓2

Loop:

My programs do not work correctly with a loop. Strings that are not palindromes fill up the stack, reducing size available for subsequent palindromes.

You could alter the program to keep track of how many times you've put a value on the stack, but this reduces the max palindrome size to 126 characters. I have included this code with the assignment, labelled "Checking Multiple Palindromes". ✓2

Economising:

I'll describe for the "Without Checking" program.

- I used the stack rather than storing values in memory as there are shorter instructions for dealing with the stack and less tidying up to do. ✓2
- Prevented duplication of the lines "mov dl, c0" and "mov [dl], al" by creating the "finish" label.
- My program does not stop in the middle of the palindrome because that would take more space in the program and reduce the size of palindrome that could be accomodated. At this size, the speed difference is not very relevant.
- The structure of my stack loop (with the push first) allowed me to eliminate 1 jump command, saving 2 bytes. Also saved me from putting a 0 on the stack later, saving another 2 or 5 bytes.

217

```
start:
        push al         ; Store input value on the stack (stores a 0 at the start)
        in 00           ; Take input
        cmp al, 2E      ; Compare input to '.'
        jnz start       ; If it's not '.', take more input
next:
        mov cl, BE      ; Store the location of the last letter in the string
loop:
        mov bl, [cl]    ; Retrieve the next letter from the end of the string
        dec cl          ; Calculate position of next last letter
        pop al          ; Retrieve the next letter from the start of the string
        cmp al, 00      ; If it's 00, we've reached the end of the string
        jz yes          ; -> It's a palindrome, jump to 'yes'
        cmp al, bl      ; Compare the two letters
        jz loop         ; If they're the same, keep checking
no:                     ; If not, it's not a palindrome
        mov al, 4E      ; Store the letter 'N'
        jmp finish      ; skip next instruction
yes:
        mov al, 59      ; Store the letter 'Y'
finish:
        mov dl, C0      ; Prepare the display
        mov [dl], al    ; Put either 'Y' or 'N' on the screen
        end
```

10

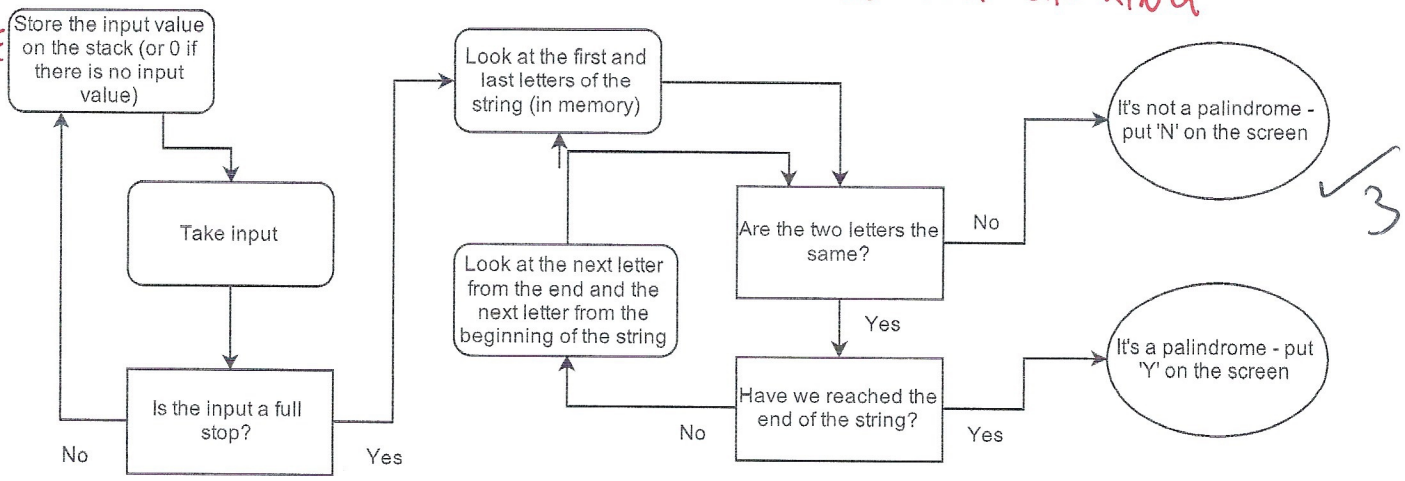
```
store:
        push al         ; Put input value on stack (puts a 0 on at the start)
input:
        in 00           ; Take input
        cmp al, 2E      ; If it's '.', stop taking input
        jz next
        cmp al, 41      ; If it's below 41 it's a symbol - ignore it
        js input
        cmp al, 5B      ; If it's below 5B it's a letter - store it
        js store
        cmp al, 61      ; If it's below 61 it's a symbol - ignore it
        js input
        cmp al, 7B      ; If it's below 7B it's a letter - store it
        js store
        jmp input       ; Otherwise ignore it
next:
        mov cl, BE      ; Save the location of the end of the string
loop2:
        mov bl, [cl]    ; Retrieve the next letter from the end of the string
        dec cl          ; Decrement end location
        pop al          ; Retrieve the next letter from the start of the string
        cmp al, 00      ; If it's 00, we've checked the whole string
        jz yes          ; -> It's a palindrome - jump to 'yes'
        cmp al, bl      ; Compare the two retrieved letters
        jz loop2        ; If they're the same check the next two letters
no:                     ; Otherwise it's not a palindrome
        mov al, 4E      ; Put 'N' in a register
        jmp finish      ; Jump to display
yes:
        mov al, 59      ; Put 'Y' in a register
finish:
        mov dl, C0      ; Prepare screen
        mov [dl], al    ; Put 'Y' or 'N' on screen
        end
```

CHECKING MULTIPLE PALINDROMES

```
begin:
        mov dl, BF          ; Initialise counter
start:
        push al
        dec dl              ; Decrement counter to follow stack pointer
        in 00
        cmp al, 2E
        jnz start
next:
        mov cl, BE
loop:
        mov bl, [cl]
        dec cl
        pop al
        inc dl              ; Increment counter to follow stack pointer
        cmp al, 00
        jz yes
        cmp al, bl
        jz loop
no:
        mov al, 4E
        jmp finish
yes:
        mov al, 59
finish:
        mov cl, C0
        mov [cl], al
        mov al, 00
loop2:
        inc dl              ; Increment counter
        cmp dl, C0          ; If you're past the stack,
        jz begin            ; Begin again
        pop al              ; Otherwise pop from the stack
        jmp loop2           ; Repeat
end
```

WITHOUT CHECKING

START HERE →

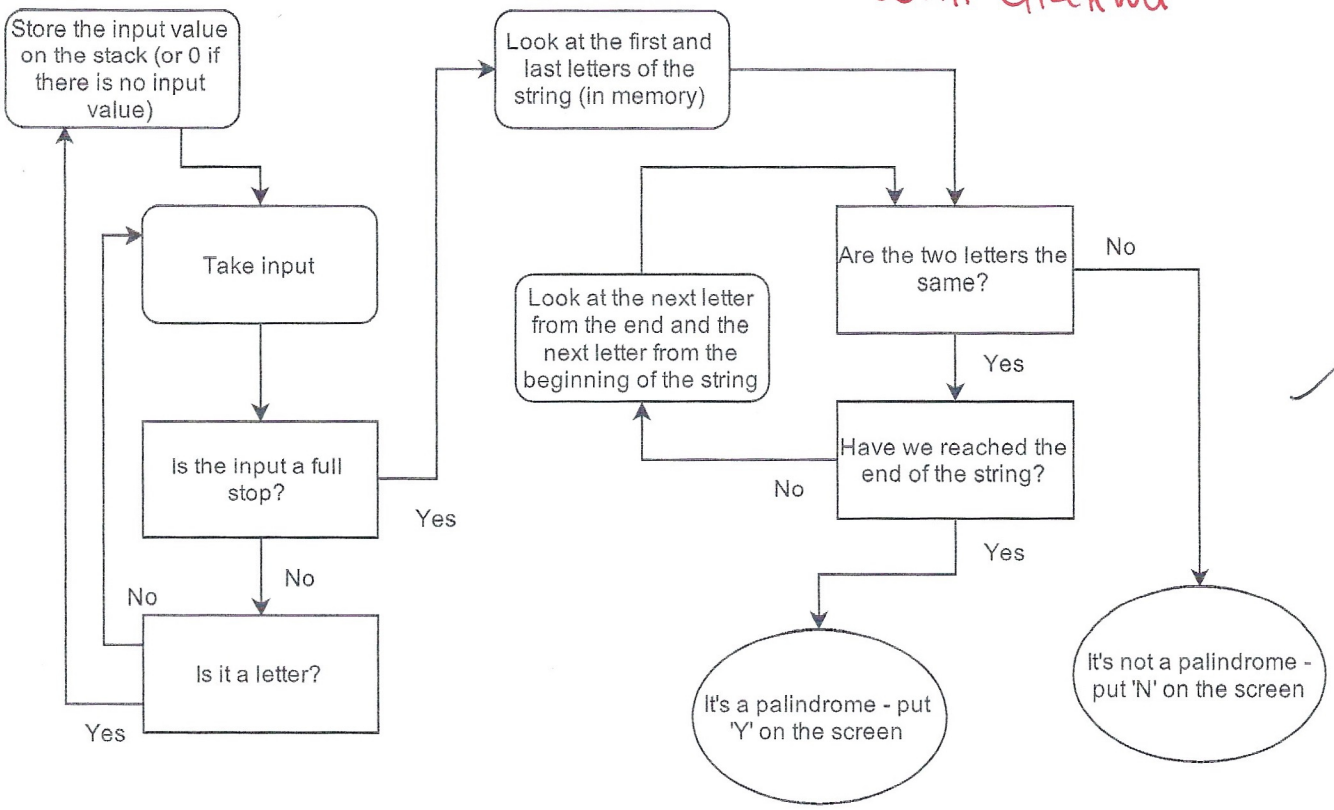


✓3

6/7

START HERE →

WITH CHECKING



7/7