CS 322-Lab 5

Name: P.V. Sriram

Roll No.: 1801CS37

# The Sorting Hat

The sorting hat game is inspired from the sorting hat in Harry potter. The Sorting hat is responsible to assign newly enrolled students into the Houses of Gryffindor, Slytherin, Ravenclaw, Hufflepuff. The hat is placed on the student’s head and based on the personality; it speaks out the best suited house. But in some cases, such as Harry Potter himself, the hat gets confused in between the best options. It couldn’t decide in between and “Slytherin” and “Gryffindor”, at the end decided to choose Gryffindor just because his father belonged to it.

We will help the Sorting hat to take its decision using an assembly level program. If there’s a confusion between Gryffindor and Slytherin.

# Algorithm

We first take the first name of the student. And then we search for the substring “ram” within the name. If there is a substring “ram” within the name, we assign Gryffindor, else Slytherin.

**Read Until Enter**: Calls system interrupt for keyboard input. We take in a maximum of 10 characters and place them in the reserved memory “buffer”.

The string “ram” is stored in a variable location “subs”.

**Find**: In this function, we loop through the characters of the first name and see for a match in the first character of the name and ram. If there is a match, we jump to eql function else, we jump to not\_eql function.

**Eql:** If there’s a match we see the next characters in both name, and ram. And do this three times for a continuous match for r, a, m by incrementing esi, edi both. If we don’t find such a match, we make.

**Not\_eql**: If the present characters don’t match, we make edi 0 and search for “r” all over.

**Passed**: Prints the string “Gryffindor” if a match is found

**Not** **Passed**: Prints the string “Slytherin” if a match is not found

Text

Description automatically generated

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# Code

%include "io.mac"

segment .data

ENTER\_KEY equ 0Dh

ESC\_KEY equ 1Bh ; exit

U\_KEY equ 48h

D\_KEY equ 50h

L\_KEY equ 4Bh

R\_KEY equ 4Dh

CHARS dw 010h

color\_red db 0CCh

color\_green db 0AAh

space db 020h

row db 0

col db 0

subs db "ram"

start db "Please enter your first name...", 0

Pass\_Okay db 0Dh , 0Ah ,"Gryffindor", 10, 0

Pass\_fail db 0Dh , 0Ah ,"Slytherin", 10, 0

segment .bss

buffer resb 10

buffB resb 1

segment .text

\_main:

mov bx, ENTER\_KEY

mov ecx, 0

mov esi, 0

PutStr start

nwln

readUntilEnter:

mov eax, 3

mov ebx, 0

mov ecx, buffer

mov edx, 10

int 0x80

PassOK:

mov edx, 10

mov esi, 0

mov edi, 0

mov ecx, 3

find:

MOV al, byte[buffer + esi]

MOV ah, byte[subs + edi]

cmp al, ah

je eql

jne not\_eql

not\_eql:

inc esi

dec edx

jz not\_passed

jnz find

eql:

inc esi

inc edi

dec ecx

jz passed

dec edx

jz not\_passed

MOV al, [buffer + esi]

MOV ah, [subs + edi]

cmp al, ah

je eql

mov edi, 0

jne find

passed:

PutStr Pass\_Okay

jmp exit

not\_passed:

PutStr Pass\_fail

jmp exit

exit:

mov ebx, 0 ; return 0 status on exit - 'No Errors'

mov eax, 1 ; invoke SYS\_EXIT (kernel opcode 1)

int 80h