RISC Project

Teams can choose from one of two programming projects:

Conversationial & Game Project:
- **wopr**: this program is inspired by the movie, wargames.

Each team must turn in a report which contains the following:

1. Cover sheet with up to 3 team members names & signatures
2. Description of the problem, enhancements, & lessons learned.
3. A flowchart of the functions: talk(), game_move(), strfind().
5. Floppy disk of the (1)-(4).
6. 5 minute demo with all members present.
Wopr: example

Here is an example of how the program should work

wopr
Shall we play a game?

Global thermonuclear War

Wouldn’t you prefer a good game of chess?

tic-tAc-Toe

X: please enter your move?

1

X |   |
---+---+---
    | O |
---+---+---
    |   |
X: please enter your move?

7

X |   |
---+---+---
O | O |   
---+---+---
X |   |

X: please enter your move?

6

X | O |
---+---+---
O | O | X
---+---+---
X |   |
Wopr: con’t

X: please enter your move? 8

```
  x | 0 |
  ----
  o | o | x
  ----
  x | x | 0
```

Draw. Game over.

Shall we play a game?

**My name is Bill Gates.**

What is your name, again?

**bill from Seattle, Wa.**

Bill, Shall we play a game?

**not now.**

logoff.
Write at least these functions *(using MIPS register conventions):*

- **main()**  
  # Main program: calls talk & game

- **talk()**  
  # 0:exit, 1:play game

- **game_print(&array)**  
  # prints the tic-tac-toe board  
  # player: 0=O, 1=X, -1=blank

- **game_init(&array)**  
  # initializes the board to blank

- **game_set(&array,position,player)**

- **game_move(&array,player)**

- **gets(char *string)**  
  # No system calls allowed

- **puts(char *string)**  
  # No system calls allowed

- **strcmp(s1,s2)**  
  # -1:s1<s2; 0:s1==S2; 1:s1>s2

- **strlower(string)**

- **strfind(string,s1)**  
  #1:s1 not found, 0:s1 found
Wopr: talk()

The talk function can be on any topic you want: Wouldn’t you prefer a good game of chess?
can become (i.e. baseball, cooking, psychology, …) Wouldn’t you prefer to talk about yourself?
I am very happy about myself. Exactly, how happy are you?
...

By using the strfind() and combining it with logical ands and logical or you can have interesting responses.

1) At least a one 3 level logical AND condition nesting is required in the program.
2) At least 3 different types of conversation pattern matching.
3) This will be graded for creativity
ANSI C Language function: `char *gets(char *s)` where char *s is a pointer to a pre-allocated string of bytes.

Gets returns the original pointer *s* passed in.

Gets inputs each character and echos it until a newline is encountered (0x0a). The newline is not saved in the final string. The returned string is null terminated.

ANSI C Language function: `int puts(char *s)` where char *s is a pointer to a string of bytes to be printed.

Puts prints each character until a null is encountered (0x0a) in the string. A newline is then also printed to the console.

Puts returns the number of characters written to the console.
Rx: Memory Mapped char i/o

IF Ready bit is true THEN there is a new data character

Receiver control status: memory address 0xffff0000

Unused Ready Bit

Receiver data: memory address 0xffff0004

Unused byte

Rx:  

li $t0, 0xffff0000
lw $t1, 0($t0) #get rx status
andi $t1, 0x0001 #ready?
beq $t1, $zero, Rx #no
lbu $v0, 4($t0) #yes - get byte
**Tx: Memory Mapped character i/o**

IF Tx Ready bit is true THEN ok to output a character

**Transmitter control status:** memory address 0xffff0008

<table>
<thead>
<tr>
<th>Unused</th>
<th>Ready Bit</th>
</tr>
</thead>
</table>

**Transmitter data:** memory address 0xffff000c

<table>
<thead>
<tr>
<th>Unused</th>
<th>byte</th>
</tr>
</thead>
</table>

**Tx:**

```
li $t0, 0xffff0008
lw $t1,0($t0) #get tx status
andi $t1,0x0001 #ready?
beq $t1,$zero,Tx #no
stb $a0,4($t0) #yes - put byte
```
Rx_line: Read a line from the console.

#Make sure -mapped_io is enabled on spim

rx_line:
    la    $s0, rx_buffer  #string pointer
    li    $t1, 0xffffffff
rx_line1:
    lw    $t2,0($t1)    # ready?
    andi  $t2,$t2,1
    beq   $t2,$0,rx_line1  #no - loop
    lbu   $t2,4($t1)   #yes - get char
    sb    $t2,0($s0)  #..store it
    addi  $t2,$t2,-10 #carrage return?
    beq   $t2,$0,rx_done #yes - make it zero
    addi  $s0,$s0,1 #next string addr
    j     rx_line1