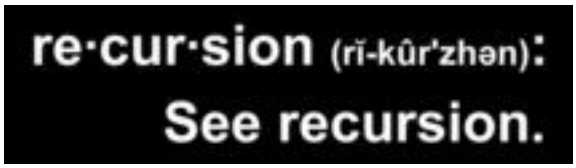
CS 218 – MIPS Assignment #5

Purpose:	Become familiar with the MIPS Instruction Set, and the MIPS standard calling convention, and basic recursion.
Due:	Tuesday (4/30)
Points:	80

Assignment:

Write a MIPS assembly language program to determine if the provided starting location and game squares represent a winnable puzzle game. The puzzle game consists of a row of squares, each containing an integer.



For example:

(3)	6	4	1	3	4	2	5	3	0

The circle on the initial square (position 0) is a marker that can move to other squares along the row. At each step in the puzzle, you may move the marker the number of squares indicated by the integer in the square it currently occupies. The marker may move either left or right along the row but may not move past either end. For example, the only legal first move is to move the marker three squares to the right because there is no room to move three spaces to the left. The goal of the puzzle is to move the marker to the 0 at the far end of the row. In this configuration, you can solve the puzzle by making the following set of moves:

start	(3)	6	4	1	3	4	2	5	3	0
move right 3	3	6	4	(1)	3	4	2	5	3	0
move left 1	3	6	(4)	1	3	4	2	5	3	0
move right 4	3	6	4	1	3	4	(2)	5	3	0
move right 2	3	6	4	1	3	4	2	5	(3)	0
move left 3	3	6	4	1	3	(4)	2	5	3	0
move right 4	3	6	4	1	3	4	2	5	3	(0)

Even though this puzzle is solvable—and indeed has more than one solution – some puzzles of this form may be impossible, such as the following one:

(3)	1	2	3	0

In this puzzle, starting from position 0, you can bounce between the two 3's, but you cannot reach any other squares. The puzzle would be winnable if the starting position was position 1.

Using the provided main, write the following MIPS assembly language functions:

- MIPS void function, ***showGame()***, to display a formatted game board. The output should include a passed game board number and a formatted row of numbers. The formatting should be as follows:

```
**********************************************************************
Game Board: 1

   ----------------------------------------
   | 3 | 6 | 4 | 1 | 3 | 4 | 2 | 5 | 3 | 0 |
   ----------------------------------------
```

  All numbers will be between 0 and 9. You may assume valid data.

- MIPS value returning function, ***getStartPosition()***, to prompt for, read, and verify a starting position. The starting position must be ≥ 0 and < length (which is passed). If correct, the function should return the start position (via reference) and return TRUE. If there are more than 3 errors, the function should print a messages and return FALSE.

- A recursive MIPS void function, ***canWin()***, to determine if the passed game board can be won from the provided starting position. The function should accept a starting position of the initial marker along with the populated games squares array, and the array length. The function should return TRUE if it is possible to solve the puzzle from the starting configuration and FALSE if is not possible. You may assume all the integers in the array are positive and > 0 except for the last entry, the goal square, which is always zero. The values of the elements in the array must be the same after calling your function as they are beforehand, which is to say if you change them during processing, you need to change them back! The general recurrence relation is:

$$canWin(pos, brd[], len) = \begin{cases} FALSE & if\ pos < 0 \\ FALSE & if\ pos \geq len \\ TRUE & if\ brd[pos] = 0 \\ canWin(pos + brd[pos]) \vee & \\ \quad canWin(pos - brd[pos]) & otherwise \end{cases}$$

  If a recursive call is made, the board for that position can be marked with a -1 (to indicate it has been accessed). You will need to restore that position to the original value when done. The ∨ is a logical OR.

- MIPS void function, ***displayResult()***, to re-display the board followed by a message showing the result (can win or can not win). The strings are provided.

- MIPS value returning function, ***doAnother()***, to prompt the user ("**`Try another start position (y/n)?`**") if they want to try another start position for that game board. If invalid input is provided, display an error message and re-prompt. *Note*, you only need to check for a lower case response.

Submission:

When complete, submit:

- A copy of the **source file** via the class web page before class time.

Example Output:

```
MIPS Assignment #5
Row Puzzle Solution Program


********************************************************************************
Game Board: 1

   -----------------------------------------
   | 3 | 6 | 4 | 1 | 3 | 4 | 2 | 5 | 3 | 0 |
   -----------------------------------------

Enter a start position (0-9): -1

Error, invalid start position. Please re-enter.
Enter a start position (0-9): 12

Error, invalid start position. Please re-enter.
Enter a start position (0-9): 0

********************************************************************************
Game Board: 1

   -----------------------------------------
   | 3 | 6 | 4 | 1 | 3 | 4 | 2 | 5 | 3 | 0 |
   -----------------------------------------

Starting from position 0 you can win!  :-)

Try another start position (y/n)? n


********************************************************************************
Game Board: 2

   ----------------------
   | 3 | 1 | 2 | 3 | 0 |
   ----------------------

Enter a start position (0-4): 0

********************************************************************************
Game Board: 2

   ----------------------
   | 3 | 1 | 2 | 3 | 0 |
   ----------------------

Starting from position 0 you can not win.  :-(

Try another start position (y/n)? y

Enter a start position (0-4): 1

********************************************************************************
Game Board: 2

   ----------------------
   | 3 | 1 | 2 | 3 | 0 |
   ----------------------

Starting from position 1 you can win!  :-)

Try another start position (y/n)? n
```

Note, continued on next page

```
**************************************************************************
Game Board: 3

   -----------------------------------------------------
   | 3 | 7 | 2 | 4 | 3 | 4 | 3 | 2 | 7 | 3 | 3 | 5 | 2 | 0 |
   -----------------------------------------------------

Enter a start position (0-13): 0

**************************************************************************
Game Board: 3

   -----------------------------------------------------
   | 3 | 7 | 2 | 4 | 3 | 4 | 3 | 2 | 7 | 3 | 3 | 5 | 2 | 0 |
   -----------------------------------------------------

Starting from position 0 you can win!  :-)

Try another start position (y/n)? n

**************************************************************************
Game Board: 4

   -------------------------------------------------------------
   | 3 | 7 | 2 | 4 | 3 | 4 | 3 | 2 | 4 | 3 | 2 | 5 | 2 | 1 | 1 | 0 |
   -------------------------------------------------------------

Enter a start position (0-15): 0

**************************************************************************
Game Board: 4

   -------------------------------------------------------------
   | 3 | 7 | 2 | 4 | 3 | 4 | 3 | 2 | 4 | 3 | 2 | 5 | 2 | 1 | 1 | 0 |
   -------------------------------------------------------------

Starting from position 0 you can win!  :-)

Try another start position (y/n)? y

Enter a start position (0-15): -3

Error, invalid start position. Please re-enter.
Enter a start position (0-15): 17

Error, invalid start position. Please re-enter.
Enter a start position (0-15): 22

Error, invalid start position. Please re-enter.
Enter a start position (0-15): 99

Sorry, too many errors.


You have reached recursive nirvana.
Program Terminated.
```