**Nim游戏**

Nim是一个简单的双人游戏，可能起源于中国。游戏中使用的计数器类型有很多种类，如石头、火柴、苹果等。游戏界面被划分为很多行，每行中有数量不等的计数器，如图1所示：

|  |  |
| --- | --- |
| 行号 | 计数器数量 |
| 1 | ○○○ |
| 2 | ○○○○○○ |
| …… | …… |
| n | ○○○○○○○○○○ |

图1 游戏界面

1. **游戏规则**

本次实验对Nim游戏做了一些小的改变，具体如下：游戏界面由三行组成，计数器类型为石头，其中A行包含3个石头，B行包含5个石头，C行包含8个石头。

规则如下：

⑴ 每个玩家轮流从某一行中移除一个或多个石头。

⑵ 一个玩家不能在一个回合中从多个行中移除石头。

⑶ 当某个玩家从游戏界面上移除最后剩余的石头时，此时游戏结束，该玩家获胜。

1. **实验要求**

⑴ 在游戏开始时，你应该显示游戏界面的初始化状态。具体包括：在每行石头的前面，你应该先输出行的名称，例如“ROW A”。你应该使用ASCII字符小写字母“o”（ASCII码 x006F）来表示石头。游戏界面的初始化状态应该如下：

*ROW A: ooo*

*ROW B: ooooo*

*ROW C: oooooooo*

⑵ 游戏总是从玩家1先开始，之后玩家1和玩家2轮流进行。在每一个回合开始时，你应该输出轮到哪一个玩家开始，并提示玩家进行操作。例如，对于玩家1，应该有如下显示：

*Player 1,choose a row and number of rocks:*

⑶ 为了指定要移除哪一行中的多少石头，玩家应该输入一个字母后跟一个数字（输入结束后不需要按Enter键），其中字母（A,B或C）指定行，数字（从1到所选行中石头的数量）指定要移除的石头的数量。你的程序必须要确保玩家从有效的行中移除有效数量的石头，如果玩家输入无效，你应该输出错误提示信息并提示该玩家再次进行输入。例如，如果轮到玩家1：

*Player 1, choose a row and number of rocks: D4*

*Invalid move. Try again.*

*Player 1, choose a row and number of rocks: A9*

*Invalid move. Try again.*

*Player 1, choose a row and number of rocks: A\**

*Invalid move. Try again.*

*Player 1, choose a row and number of rocks: &4*

*Invalid move. Try again.*

*Player 1, choose a row and number of rocks:*

你的程序应保持提示玩家，直到玩家选择有效的输入为止。确保你的程序能够回显玩家的输入到屏幕上，当回显玩家的输入后，此时应该输出一个换行符（ASCII码x000A）使光标指向下一行。

⑷ 玩家选择有效的输入后，你应该检查获胜者。如果有一个玩家获胜，你应该显示相应的输出来表明该玩家获胜。如果没有胜利者，你的程序应该更新游戏界面中每行石头的数量，重新显示更新的游戏界面，并轮到下一个玩家继续。

⑸ 当某个玩家从游戏界面上移除最后的石头时，游戏结束。此时，你的程序应该显示获胜者然后停止。例如，如果玩家2移除了最后的石头，你的程序应该输出一下内容：

*Player 2 Wins.*

1. **样例输入/输出**

**注意：你的程序中输入输出的格式必须完全和样例中的格式相一致**

ROW A: ooo

ROW B: ooooo

ROW C: oooooooo

Player 1, choose a row and number of rocks: B2

ROW A: ooo

ROW B: ooo

ROW C: oooooooo

Player 2, choose a row and number of rocks: A1

ROW A: oo

ROW B: ooo

ROW C: oooooooo

Player 1, choose a row and number of rocks: C6

ROW A: oo

ROW B: ooo

ROW C: oo

Player 2, choose a row and number of rocks: G1

Invalid move. Try again.

Player 2, choose a row and number of rocks: B3

ROW A: oo

ROW B:

ROW C: oo

Player 1, choose a row and number of rocks: A3

Invalid move. Try again.

Player 1, choose a row and number of rocks: C2

ROW A: oo

ROW B:

ROW C:

Player 2, choose a row and number of rocks: A1

ROW A: o

ROW B:

ROW C:

Player 1, choose a row and number of rocks: A\*

Invalid move. Try again.

Player 1, choose a row and number of rocks: &4

Invalid move. Try again.

Player 1, choose a row and number of rocks: A1

Player 1 Wins.

----- Halting the processor -----

1. **提示与建议**

⑴ 记住，程序中所有的输入输出使用ASCII字符，你应该负责进行必要的转换。

⑵ 从键盘中输入字符你应该使用TRAP x20（GETC）指令，同时为了回显输入的字符到屏幕上，你应该使用TRAP x21（OUT）指令，该指令紧跟在TRAP x20指令之后。

⑶ 你应该在适当的时候使用子程序。

⑷ 在你编写的每个子程序中，应该保存并还原所使用的任何寄存器。这将避免你在调试过程中遇到问题。

⑸ 在一个回合中，玩家的输入必须包含指定为A,B或C（即大写字母）的行，后面紧跟不大于该行仍然存在的石头数量的数字。

**提示：**

① 你应该设置程序的开始地址在x3000（如,程序的第一行指令应该为 .ORIG x3000）

② 源文件命名为**nim.asm**

**.orig x3000**

**again jsr print**

**jsr datain1**

**jsr print**

**jsr datain2**

**br again**

**print st r0,save\_r0**

**st r1,save\_r1**

**st r7,save\_r7**

**lea r0,row\_a**

**puts**

**ld r0,stone**

**ld r1,num\_a**

**loop\_a out**

**add r1,r1,#-1**

**brp loop\_a**

**ld r0,cr**

**out**

**lea r0,row\_b**

**puts**

**ld r0,stone**

**ld r1,num\_b**

**loop\_b out**

**add r1,r1,#-1**

**brp loop\_b**

**ld r0,cr**

**out**

**lea r0,row\_c**

**puts**

**ld r0,stone**

**ld r1,num\_c**

**loop\_c out**

**add r1,r1,#-1**

**brp loop\_c**

**ld r0,cr**

**out**

**ld r0,save\_r0**

**ld r1,save\_r1**

**ld r7,save\_r7**

**ret**

**save\_r0 .fill #0**

**save\_r1 .fill #0**

**stone .fill x006f**

**cr .fill x000d**

**row\_a .stringz "ROW A: "**

**row\_b .stringz "ROW B: "**

**row\_c .stringz "ROW C: "**

**num\_a .fill #3**

**num\_b .fill #5**

**num\_c .fill #8**

**cue1 st r0,save\_r0**

**st r7,save\_r7**

**lea r0,play1**

**puts**

**ld r0,save\_r0**

**ld r7,save\_r7**

**ret**

**play1 .stringz "Player 1,choose a row and number of rocks:"**

**cue2 st r0,save\_r0**

**st r7,save\_r7**

**lea r0,play2**

**puts**

**ld r0,save\_r0**

**ld r7,save\_r7**

**ret**

**play2 .stringz "Player 2,choose a row and number of rocks:"**

**save\_r7 .fill #0**

**datain1 st r0,save\_r0**

**st r2,save\_r2**

**st r3,save\_r3**

**st r7,saver7**

**try1 jsr cue1**

**getc**

**out**

**add r2,r0,#0**

**not r2,r2**

**add r2,r2,#1**

**getc**

**out**

**add r3,r0,#0**

**ld r0,lf**

**out**

**test1a ld r0,char\_a**

**add r0,r2,r0**

**brnp test1b**

**ld r0,char\_0**

**not r0,r0**

**add r0,r0,#1**

**add r0,r0,r3**

**brn error1**

**ld r3,num\_a**

**not r0,r0**

**add r0,r0,#1**

**add r3,r0,r3**

**brn error1**

**st r3,num\_a**

**ld r3,sum\_abc**

**add r3,r3,r0**

**brz win1**

**st r3,sum\_abc**

**br save**

**test1b ld r0,char\_b**

**add r0,r2,r0**

**brnp test1c**

**ld r0,char\_0**

**not r0,r0**

**add r0,r0,#1**

**add r0,r0,r3**

**brn error1**

**ld r3,num\_b**

**not r0,r0**

**add r0,r0,#1**

**add r3,r0,r3**

**brn error1**

**st r3,num\_b**

**ld r3,sum\_abc**

**add r3,r3,r0**

**brz win1**

**st r3,sum\_abc**

**br save**

**test1c ld r0,char\_c**

**add r0,r2,r0**

**brnp error1**

**ld r0,char\_0**

**not r0,r0**

**add r0,r0,#1**

**add r0,r0,r3**

**brn error1**

**ld r3,num\_c**

**not r0,r0**

**add r0,r0,#1**

**add r3,r0,r3**

**brn error1**

**st r3,num\_c**

**ld r3,sum\_abc**

**add r3,r3,r0**

**brz win1**

**st r3,sum\_abc**

**br save**

**win1 ld r0,lf**

**out**

**lea r0,wins1**

**puts**

**halt**

**error1 lea r0,invalid**

**puts**

**ld r0,cr2**

**out**

**br try1**

**datain2 st r0,save\_r0**

**st r2,save\_r2**

**st r3,save\_r3**

**st r7,saver7**

**try2 jsr cue2**

**getc**

**out**

**add r2,r0,#0**

**not r2,r2**

**add r2,r2,#1**

**getc**

**out**

**add r3,r0,#0**

**ld r0,lf**

**out**

**test2a ld r0,char\_a**

**add r0,r2,r0**

**brnp test2b**

**ld r0,char\_0**

**not r0,r0**

**add r0,r0,#1**

**add r0,r0,r3**

**brn error2**

**ld r3,num\_a**

**not r0,r0**

**add r0,r0,#1**

**add r3,r0,r3**

**brn error2**

**st r3,num\_a**

**ld r3,sum\_abc**

**add r3,r3,r0**

**brz win2**

**st r3,sum\_abc**

**br save**

**test2b ld r0,char\_b**

**add r0,r2,r0**

**brnp test2c**

**ld r0,char\_0**

**not r0,r0**

**add r0,r0,#1**

**add r0,r0,r3**

**brn error2**

**ld r3,num\_b**

**not r0,r0**

**add r0,r0,#1**

**add r3,r0,r3**

**brn error2**

**st r3,num\_b**

**ld r3,sum\_abc**

**add r3,r3,r0**

**brz win2**

**st r3,sum\_abc**

**br save**

**test2c ld r0,char\_c**

**add r0,r2,r0**

**brnp error2**

**ld r0,char\_0**

**not r0,r0**

**add r0,r0,#1**

**add r0,r0,r3**

**brn error2**

**ld r3,num\_c**

**not r0,r0**

**add r0,r0,#1**

**add r3,r0,r3**

**brn error2**

**st r3,num\_c**

**ld r3,sum\_abc**

**add r3,r3,r0**

**brz win2**

**st r3,sum\_abc**

**br save**

**win2 ld r0,lf**

**out**

**lea r0,wins2**

**puts**

**halt**

**error2 lea r0,invalid**

**puts**

**ld r0,cr2**

**out**

**br try2**

**save ld r0,lf**

**out**

**ld r0,saver0**

**ld r2,save\_r2**

**ld r3,save\_r3**

**ld r7,saver7**

**ret**

**lf .fill x000a**

**char\_a .fill x0041**

**char\_b .fill x0042**

**char\_c .fill x0043**

**char\_0 .fill x0030**

**wins1 .stringz "Player 1 Wins."**

**wins2 .stringz "Player 2 Wins."**

**invalid .stringz "Invalid move. Try again."**

**cr2 .fill x000d**

**sum\_abc .fill #16**

**saver0 .fill #0**

**save\_r2 .fill #0**

**save\_r3 .fill #0**

**saver7 .fill #0**

**.end**