LAB 04: The Game of Nim

Nim is a simple two-player game. There are many variations of this game with respect to the type of counters used (stones, matches, apples, etc.), the number of counters in each row, and the number of rows in the game board.

Rules

In our variation of Nim, the game board consists of three rows of rocks. Row A contains 3 rocks, Row B contains 5 rocks, and Row C contains 8 rocks.
The rules are as follows:

1. Each player takes turns removing one or more rocks from a single row.

2. A player cannot remove rocks from more than one row in a single turn

3. The game ends when a player removes the last rock from the game board. The player who removes the last rock loses.

What to do

✧ At the beginning of the game you should display the initial state of the game board. Before each row of rocks you should output the name of the row, for instance "Row A:". You should use the ASCII character lowercase "o" (ASCII code x006F) to represent a rock. The initial state of the game board should look as follows:

ROW A: ooo

ROW B: ooooo

ROW C: oooooooo

✧ Player 1 always goes first, and play alternates between Player 1 and Player 2. At the beginning of each turn you should output which players turn it

is, and prompt the player for her move. For Player 1 this should look as follows:

Player 1, choose a row and number of rocks:

- To specify which row and how many rocks to remove, the player should input a letter followed by a number (they do NOT need to press Enter after inputting a move). The letter (A, B, or C) specifies the row, and the number (from 1 to the number of rocks in the chosen row) specifies how many rocks to remove. Your program must make sure the players move has a valid row and number of rocks. If the players move is invalid, you should output an error message and prompt the same player for a move. For example, if it is Player 1s turn:

  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:

- Your program should keep prompting the player until a valid move is chosen. Be sure your program echoes the players move to the screen as they type it. After you have echoed the players move, you should output a newline character (ASCII code x000A) to move the cursor to the next line.

- ✧ After a player has chosen a valid move, you should check for a winner. If there is one, display the appropriate banner declaring the winner. If there is no winner, your program should update the state of the game board to reflect the move, re-display the updated game board, and continue with the next players turn.

- ✧ When a player has removed the last rock from the game board, the game is over. At this point, your program should display the winner and then halt. For example, if Player 2 removes the last rock, your program should output the following:

  Player 1 Wins.

Sample Input/Output

An example of the input/output for a game being played can be found **here**. **To receive full credit for your program, your input/output format MUST EXACTLY MATCH the format in the example.**

Notes and Suggestions:

1. Remember, all input and output functions use ASCII characters. You are responsible for making any conversions that are necessary.

2. For character input from the keyboard in this assignment, you should use TRAP x20 (GETC). To echo the characters onto the screen, you should follow each TRAP x20 with a TRAP x21 (OUT). Recall that TRAP x23 displays a banner to prompt the person at the keyboard to input a character. You do not need that banner since your program has its own style of prompt. Therefore you should use TRAP x20 which does the same as TRAP x23 except it does not print a banner on the screen to prompt for input.

3. You should use subroutines where appropriate.

4. In each subroutine you write, you should save and restore any registers that you use. This will avoid a major headache during debugging.

5. A legitimate turn must contain the row, specified as A, B, or C (i.e., capital letter) followed by a number that is not larger than the number of rocks still remaining in that row.

Additional Requirements:

If you don't comply with these requirements, the lab may be counted as an invalid work.

1. The report shall contain at least 3 parts: How do you work out the algorithm? How do you write the program? And how do you design your own test cases to ensure the program works fine?
2. Save your report in pdf format and name it such as **report.pdf**.
3. Your program should be renamed to **nim.asm.**
4. Put all above in a directory named after your student number and pack it **(e.g. PB07210340_张海博_LAB04.zip).**