CS330 - Computer Organization & Assembly Language
Assignment #7 (Sokoban)

*****Individual Work Only*****

ASSIGNMENT:

Create a working Sokoban game.

For an overview of Sokoban, see this link: https://en.wikipedia.org/wiki/Sokoban

A free online version of the game can be found here:
https://www.mathsisfun.com/games/sokoban.html

The rules of Sokoban are quite simple. The goal is to push all the stars onto the goal squares. The player can only move in the four cardinal directions (up, down, left, right), and can only push the star (cannot pull). The player cannot move or push stars into walls (or stars into other stars).

To implement the game we will create a loop inside our main() function, and implement the following two functions: **validMove()**, and **movePlayer()**

An overview of the project is provided in a YouTube video here: https://youtu.be/UlTzsuR2SgM
Recommend watching the video, at least the beginning and end which covers how to download the Starter-code and a code walk-through.

Starter code is provided on Canvas to facilitate.

Also, a .pdf is provided to describe the game, rules, approach, helper functions, and starter code.

Be sure to review the video, starter code, and .pdf before beginning.

To obtain the bonus credit, complete the Sokoban game (**see #3 below**).

Be sure to use the map representation, and player struct as described in the code.

Be sure to use pointer arithmetic to retrieve and assign map values

1. To Begin, either:
    a. Download the .zip file from Canvas, copy to Vulcan (or Codespaces), and unzip. Vulcan detailed instructions follow:
        i. Download the zip file to your computer
        ii. Open up a Terminal
        iii. Navigate to the folder on your computer where you stored the zip file
        iv. If you're not on UABSecure Network, open the VPN
        v. Copy the file to Vulcan using Secure Copy (SCP) or Secure File Transfer Protocol (SFTP). Using SCP the command is (see also Command Line Video):
            1. scp sokoban.zip <BlazerID>@moat.cis.uab.edu:~/

2. Enter your password when prompted

vi. Unzip the file:

1. `unzip sokoban.zip -d sokoban`

b. Or Clone the repository from GitLab

i. If you're not on UABSecure Network, open the VPN

ii. Open a Terminal and ssh onto Vulcan

1. You can do this in the VSCode Terminal
2. Or you can open a separate terminal and use the following command (see Command Line Video for more details):
    a. ssh <BlazerID>@moat.cis.uab.edu

iii. Once on Vulcan, use the following command to clone the repository (the following should be all on one line):

```
git clone
https://gitlab.rc.uab.edu/bedingjd/cs330_sokoban.git
```

2. Once the files are on Vulcan, navigate to your new sokoban folder (with the files), and:
    a. `make`
    b. `make run`
    c. The game should run correctly if you have all the files
3. **Modify the file cs330_sokoban_game.c**
    a. **Create the two functions validMove() and movePlayer()**
    b. **Modify the main() function accordingly**
    c. **Test**
4. Submit the completed program to Canvas as outlined in the Turn-in Section below

TURN IN:

- cs330_sokoban_game.c containing all functions, which compiles and runs on Vulcan (or Codespaces), and is documented appropriately
- a Makefile, where (a) the first rule compiles your code, and (b) there is a rule called "run" that will run the code. Therefore, the command "make" will compile the code, and the command "make run" will run the code.
- Independent completion form
- The original files:
    - libsok_helper_vulcan.a
    - sok_header.h
    - maps.txt
- All files should be inside a .zip file

RUBRIC:

- 80%: 30% for each working function and 20% for the main() function
    - The player structure provided should be used
    - Pointer arithmetic should be used to access and write map values
- 20%: Document all code

    - Comments must be thorough and correct but not redundant
    - Explain both what you're doing and why you're doing it.
- NOTE:
    - Code that does not compile on Vulcan (or Codespaces) using the Makefile will result in an automatic zero.
    - Submissions not in a .zip file, missing a Makefile, or missing an Independent Completion form will result in an automatic zero.