The Problem

We're going to implement a program to play solitaire. Instead of traditional Klondike, however, we're going to work with a somewhat easier to program version. The way the game is played, is that the pack of 52 cards is shuffled, and then used to deal the Tableau. The tableau is the cards in play on the table. The remaining cards after the tableau is dealt out form the stock. The tableau is formed by placing four cards in four piles (total of 16 cards), face down. The top card of each pile is then turned up. A single card is placed face up to the right of the topmost pile; this is called the starter.

The rules of the game are as follows: If a card on the top of a pile is of the same suit as the starter, move it next to the starter. If a card is the same rank as the starter, move it below the starter, next to the second pile from the top. It forms the starter of the second suit. When you cannot move any more cards in this manner, begin turning cards from the stock, 3 at a time. If
three cards are not available, flip the remaining cards, and turn the pack over for the next flip. If a card is of one of the same suit as the starter then you may place it next to it's pile. If a card is the same rank as the original starter, you may start a new suit. If a card is of a "started" suit, besides the first, you may only move it into the tableau if the card of the same rank appears in the suit above it.

Write a program to allow a user to play this game. It should have a simple text interface which displays the piles; a tab and then the values of the cards in the started suit. Below that, it should show the top of the stock.

Solitare (C) Kenneth Flynn

AD 5C, 6C, 3C
2H 5D
KS
3H

Stock: (4 remaining) 5S
Action: 1] Play card from stock
        2] Play card from pile
        3] Turn next stock
        4] Quit

Choice? _

For each action, you should check to see if it is legal, and if so, make it. For playing a card from a pile, you should prompt for which pile.

Hints

- **Warning**: This project is very difficult. It is intended to be a challenge for all students. Begin work EARLY! You may not be able to get this project fully working. If not, you will be graded on what you do complete.
- Where to begin? Write an algorithm for this game. The first task is to shuffle the deck. Construct a deck, in order, of all the legal cards in the game. It is probably best to represent the deck as numbers from 1 to 52, where the number / 4 = the suit, and the number % 4 is the value of the card (note that 1 = Ace, 11 = Jack, 12 = Queen, 13 = King). Once you've constructed the deck, shuffle it by picking two random cards, and switching them. Do this a large number of times (maybe 500). You will probably want to use a function of two here.
- Once you've shuffled the deck, deal the tableau (which is probably several arrays: 1 each for the piles, 1 each for the played cards, and one for the stock.
- Loop until the user quits or wins the game (plays all the cards from the stock and from the piles. You'll need to keep track of how many cards are in each.
- Back up your files before you begin working. It's just a good idea. Call your backups something like solitaire.bak.
- An executable version of code to solve this program will be made available on UNIX and PC stations, so you can get a feel for the kind of interaction the program should have. Also, if needed, we will hold a special help session for this project. A demonstration of the game will also be made available upon request.
Submission & Grading

You may use any compiler you have access to run your project. Make sure that it is strictly ANSI compatible. (I suggest compiling it on RCS using "gcc -ansi -Wall -pedantic" as suggested in lecture #1 -- that's how we will grade it).

Create a file called README, containing the following information: your name, where you developed the project, your compiler, instructions on your program's use, and anything else you think we should know in grading your project.

To submit your project, transfer it to your RCS account. If you've developed it in Sage 3101 or another PC lab, copy your files to disk and bring them to the lab. Use FTP Voyager to connect to RCS, and transfer the files. Name your project file "craps.c" At a UNIX prompt, execute the following commands:

```
tar cvf project2.tar solitare.c README
gzip project2.tar
uuencode project2.tar.gz project2.tar.gz > project2.tar.gz.uu
mail lij3@cs.rpi.edu < project2.tar.gz.uu
```

What you've just done is to "tar" the files, compressed them, changed them to a form suitable for electronic mail, and then mailed them to the TA.

Grading will be based on the following:
- Documentation: 20%
  - Is your program well commented?
  - Is the README file complete?
- Style: 40%
  - Is your program easy to follow logically?
  - Did you choose appropriate variable names?
  - Did you make appropriate use of whitespace and indentation?
- Functionality: 40%
  - Does your program do what it is supposed to?
  - Does your program compile correctly?