CS 2250 Final Project
La Belle Lucie

Implement a console-based version of the solitaire game La Belle Lucie. Name your program “labelle.”

Rules

Deck
One 52-card deck (no jokers).

Tableau
17 fans initially containing three cards each and one “fan” initially with one card. All cards are face up.

Foundations
Four piles, initially empty, that build ace to king in suit.

Play
Tableaus build down in suit. Empty tableau piles may not be filled. The top card of each tableau fan may be moved to other tableau piles or to foundation piles.

Shuffles
The tableau cards may be shuffled and redealt twice during play.

The draw
Following the second (i.e., the last) shuffle, the player may select any card in the tableau (even if it is not the top card in a pile) and play it normally. Not required but see “Extra Credit” below.

Winning
The player wins when all of the cards are moved to a foundation pile.

Hints
• Search for cards that are not playable and try to uncover them by moving cards between tableau fans
• Look for cards in the tableau that cannot be moved and build on them; e.g., the king of clubs
• A card under a king is not available until the king is moved to the foundation; if a card under a king is the same suit as the king, then that suit is blocked at the point of the card trapped under the king (e.g., the 8 of clubs)
• Other patterns also result in a block: e.g., the 10 of spades is blocked
• Try not to create blocked piles
Implementation

```plaintext
<---------------------------------- tableau --------------------------------> <foundation>

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
-----------------------------------------------------
5H KS 6C JC AD TC 2D 7H 7D TD 8C 7S QD 5C AS 3C 2S 6S
4S KD QH 9S JH JD 5D 7C AH 3D 4C 6H 3S JS KC 3H 8S
4H KH 6D QS 9C 4D 8D 5S 2C 9D 9H TS 8H QC 2H AC TH

2 shuffles remaining (-1 to shuffle, -2 to end).
Move starting column:

Each column in the tableau shown above represents a fan. Cards at the bottom of a column (i.e., at the
top of a fan) are available for play - they may be moved to other tableau columns or to the foundation.
Cards are moved by entering the column number from which the card is moved and the column number
to which the card is moved. A -1 and a -2 shuffles the deck and ends the game respectively.

Data Structures

The example data structures all contain a pointer to a string. These may be modified to point to a
structure (for example, a card) or they can be generalized to point to anything. In the latter case, the
data may be of type void*:

typedef struct list
{
  void*     data;
  struct    list    link*;
}LIST;

If this is done, then the data type of the insert argument becomes void*, and the return type of the
variously named remove and peak functions also becomes void*. This latter modification necessitates
that the value returned from remove or peak must be cast to an appropriate type. For example:
(CARD *) peak(tableau[i], j).
Validating Moves

It is easiest to implement the move function, which validates and carries out user-entered moves, using a strategy called “case analysis.” That is, start with the easiest requirements or cases and work toward the most complex. At any time that a move fails one of the cases, return from the function. This also simplifies the coding of subsequent validation steps or cases. This is most easily illustrated by example.

If the play function handles the cases in which a -1 or a -2 is entered, then the move function begins by testing the value of the starting column: if it is less than 0 or greater than 17, the move is invalid and the function can safely return. If the starting column is within the correct range, then test the ending column: if it is less than 0 or greater than 21, then the move is invalid. If the move passes these two tests, then subsequent cases may safely use these values to index into arrays or search lists by position without the potential of indexing an array out of bounds. Such a step-by-step approach is important when users can enter incorrect information.

1. $0 \leq \text{start column} \leq 17$
2. $0 \leq \text{end column} \leq 21$
3. There is a card in the start column
4. If there is NOT a card in the end column, then
   a. the end column is part of the foundation: $\text{end} > 17$ (step 2 guarantees that it is not too large), and
   b. the card being moved (start column) is an ace
5. The card being moved (start column) and the card already at the destination (end column) are the same suit
   a. if the destination is in the tableau ($\text{end} \leq 17$) the ranks build down (the rank of the moved card equals the rank of the card on the top of the stack - 1)
   b. if the destination is in the foundation ($\text{end} > 17$) the ranks build up (the rank of the moved card equals the rank of the card on the top of the stack + 1)
6. Carry out the move

Extra Credit

• (10 points) On the final shuffle, La Belle allows the player to draw and play one card even if it is not at the top of its fan or stack. You may implement this rule for extra credit.

• (5 points) Writing “bullet proof” code can be challenging but is nevertheless important. One aspect of writing such code is validating user input and not crashing if the user enters incorrect data. For example, the current version of the project reads integers as its input and will fail if the user enters non-numeric data. A typical way of solving this problem is to read all data as a string. The string can be tested to insure that it only contains appropriate characters. If the string passes this test, it is then converted to the final data type, which is sometimes called “in-core” conversion and can be done in many ways.

• The sscanf function is similar to scanf or fscanf. However, the first argument is a string and the data is read from this string rather than from a file (which includes the keyboard):
  ```c
  sscanf(char* string, char* control, var, ...);
  ```
• See page 459
• See page 461
• See section 9.3