Assignment 1

BASH SHELL PROGRAM:
   -write a bash program to play a game called LINUX (much like BINGO)

FILE Assign1Updates.txt:
   -is considered part of this assignment.
   -has answers to some questions about this assignment
   -has modifications, clarifications, etc., to this assignment
   -will change as questions arise. It may even change shortly
    before the assignment is due, so check it often.

GROUPS:
   -you may work in groups of 1-4 students.
   -group members may be from different sections.

SUBMIT:
   -only ONE group member submits the assignment
   -submit your assignment using submit program submit-cps393dwoit
   -the submit program takes one argument, which is the name of
    the program you are submitting, as in:
    > submit-cps393dwoit LINUX
   -for each group member, include a comment line at the top of
    LINUX. Each line contains:
    CS (moon) userid, Last name, First name, student ID number, Section
   -if you forget to include any group member in the comments at
    the top of LINUX, ALL group members receive a 50% reduction in
    their assignment grade.

THE IDEA:

User runs LINUX with a Card (basically, a 5x5 matrix of integers).
LINUX "calls" (provides) random numbers one-by-one. Each time a number
is called, if that number appears on the user's Card, it is "marked".
User wins when a row, or column, or all 4 corners, becomes marked.

LINUX CARD:
A LINUX Card has 5 columns of 5 numbers each.
The number in the middle must ALWAYS be zero (gets marked for free).
column 1 contains 5 unique integers in [01-15]
column 2 contains 5 unique integers in [16-30]
column 3 contains 4 unique integers in [31-45] plus middle integer 00
column 4 contains 5 unique integers in [46-60]
column 5 contains 5 unique integers in [61-75]
Card numbers must have exactly 2 digits, and be separated by one space,
with no extraneous whitespace, not even at the start or end of a line.
e.g. of a LINUX Card:
12 23 42 55 74
04 19 34 46 72
07 17 00 51 69
11 30 44 56 62
09 27 40 47 67

LINUX PROGRAM INPUT:
User supplies input in A FILE, whose name is sent as an argument
to the program ($1).
Input file contains: a seed number (an integer, starting in column 1,
with no extraneous whitespace around it), followed by 25 numbers
comprising the LINUX Card (arranged as a matrix, as above).
The input file must have exactly 6 lines.
e.g., you might run your game as follows:
> LINUX input1
where file input1 contains the 6 lines:

1063
12 23 42 55 74
04 19 34 46 72
07 17 00 51 69
11 30 44 56 62
09 27 40 47 67

PLAY THE GAME ALONE:
> LINUX myInputFile
LINUX will display the list of called numbers so far followed by the
marked Card (initially, call list is empty and only 00 is marked.)
LINUX always displays Card with column titles "L", "I", "N", "U", "X".
Then, numbers in [01-75] are called until you WIN (which stops program).
User triggers the next call by entering any character. Called numbers
are printed with an appropriate prefix of "L", "I", etc. e.g., I33, X70.
If that called number is in the Card, the number is "marked" on the Card.
LINUX displays a marked number by printing "m" after it (no whitespace
between number and "m").
Each time user triggers a new call, LINUX clears the screen, and displays
the call list followed by the marked Card.
User may quit LINUX at any time, by entering character "q" (any other
character triggers another call).

PLAY THE GAME WITH OTHERS:
2 or more people may play, but they need some extra means of
communication (so they can coordinate).
Each player runs LINUX with the SAME SEED, but a different Card.
Players coordinate entering characters at the same time.
When one player wins, this player must alert the others.

WINNING:

User wins when their Card has one of these winning conditions:
  all 5 numbers in a column are marked
  all 5 numbers in a row   are marked
  all 4 numbers in the corners are marked
When a Card wins, WINNER is printed below the final displayed marked
Card, and LINUX terminates.

CALLED NUMBERS:
Are in [01,75].
Are displayed by LINUX with appropriate prefix, e.g., L09, I30.
Are unique (no repeats).
Are obtained using bash variable $RANDOM

BASH'S $RANDOM VARIABLE:
$RANDOM is an environment variable which contains a new random number
each time it is accessed. An example of using it:
> echo $((1 + $RANDOM % 10)) #a number from 1-10 inclusive
The random stream may be initialized by setting a "seed" value, by
assigning an integer to RANDOM, e.g.,
> RANDOM=3758
If two runs of LINUX both use the same seed, both runs will get
the SAME stream of random numbers, and thus the same call list.

EXIT STATUS:
Incorrect input file causes LINUX to exit before playing the game; it
MUST send these messages to STDERR and EXIT with these codes:
input file doesn't exist, is not readable, etc.:
  exit 1. "input file missing or unreadable"
input file does not have exactly 6 lines:
  exit 2. "input file must have 6 lines"
seed line is incorrect (contains one or more non-digit characters):
  exit 3. "seed line format error"
card has incorrect layout and/or number(s):

  exit 4. "card format error"
If LINUX finishes because user quits prematurely (enters q), or wins:
  exit 0.
If you discover other error conditions, have your program handle them
as above. If the error fits into one of those above, use that above
code and message. If not, use codes 5, 6, etc., and your own
appropriate error message to stderr.

FUNCTIONS:
Use functions to make your code more readable and modular. Function
names should indicate their purpose. A function's local variables
should be declared using "local".
We will not answer questions such as "how many functions do I need?", "can one function call another?", etc. You must figure
these out.

DOCUMENTATION:
Include ALL your group members' info at the top of LINUX:
 -CS (moon) userid, Last name, First name, student ID number, Section
Limit your comments. Only document tricky, or non-obvious, code.
If a function requires arguments, then include a comment saying
what arguments the function expects.

OTHER:
LINUX may use temp files, but must remove them before termination.
File Assign1Video.mp4 shows a user playing LINUX with input file named goodInput0
Directory testfiles contains some good and bad input files. It also
contains a shell program to test LINUX with some input files.

Obviously, this shell program won't run correctly, because it runs
program LINUX, which is not included.
File maybeUsefulCommands contains a list of not-obvious commands you might find useful.

MARKING:
These values are approximate.
30% correctly checking Card layout and/or number(s) (exit 4)
10% correctly checking other input file attributes (exits 1,2,3)
60% playing the game (other than checking input above)

psztwier@thebe:/usr/courses/cps393/dwoit/assignments/assign1$