Homework 4

Due: November 25, 2019 **Points:** 100

You can turn this in late, until Monday, December 2 (the day school resumes after Thanksgiving), for 80% of your score (that is, a 20% deduction).

UNIX/Linux Questions

Please put the answers to these two questions in a text file called *linux.txt* or a PDF file called *linux.pdf*.

1. (*20 points*) On December 25, 2019, at 00:00:00 PST, how many seconds since January 1, 1970, 00:00:00 UTC, have passed? (*Hint*: look at the program *date*(1).)

2. (*20 points*) How do I find the login names and times of the people currently logged into the system"pc12.cs.ucdavis.edu", and also where they logged in from?

3. (*10 points*) How do I delete a file named *-i*?

C Programming

4. (*50 points*) We will do the next step in the Game of Life program we began in Homework 2. For reference, the program *life2.c* in the directory */home/bishop/hw4-programs* reads in the file given on the command line and draws it and the border. You can use that, or the one you did for homework 3.

   The next step is to transform the board into the next generation. Each tranformation is a step. Recall the rules are as follows. Each square on the board may be populated (indicated by an 'O') or unpopulated (indicated by an 'X'). The neighbors of a cell are the cells that adjoin it vertically, horizontally, or diagonally. At each step, the following transformations are applied to *all* the cells.

   (a) If a populated cell adjoins less than 2 other populated cells, or more than 3 populated cells, it becomes unpopulated.
   (b) If a populated cell adjoins 2 or 3 populated cells, it remains populated.
   (c) If an unpopulated cell adjoins *exactly* 3 populated cells, it becomes populated.

   Your program will take the file with the predetermined pattern as its only argument. It is to print out the initial board (generation 0), and run for 50 generations (steps), printing out the board every 5 generations (so it should print generations 0, 5, 10, . . . , 50).

   As an example, given the input file containing:

```
XOX
XOX
XOX
```

   the output boards would be (ellipses indicate omitted output):

```
Generation 0:
-----
|XOX|
|XOX|
|XOX|
-----
Generation 5:
-----
|XXX|
|OOO|
|XXX|
-----

...
```

```
Generation 50:
-----
|XOX|
|XOX|
|XOX|
-----
```

As before, the board dimensions will never be more than 100 rows and 100 columns.

The output is to be written to the standard output. If there are no problems, your program is to return an exit status code of 0; otherwise, it is to return an exit status code of 1. Successful output looks as above. The following are the error messages:

- If *malloc*(3) fails, use *perror*(3) to print the message. If, for example, the return value of malloc is stored in `p`, then you would do the test as:

  ```
  if (p == NULL) perror(s);
      ...
  ```

  where `p` holds the return value of *`malloc`* and `s` is the string you are saving in the allocated space.
- If there are not exactly 2 command-line arguments (the program name and the file containing the pattern for the board), use an *`fprintf`* of the following form:

  ```
  fprintf(stderr, "Usage: %s board_pattern\n", argv[0]);
  ```
- If opening the file fails, use *`perror`* to print an error message, like this:

  ```
  perror(fname);
  ```

  where `fname` is the name of the file that you were trying to open.
- If a line in the file is of the wrong length, use an *`fprintf`* of the following form:

  ```
  fprintf(stderr, "%s: line %d is wrong length\n", fname, lineno);
  ```

  where `fname` is the name of the file and `lineno` is the line number of the bad line.
- If a line in the file contains an invalid character, use an *`fprintf`* of the following form:

  ```
  fprintf(stderr, "%s: bad character '%c' in line%d\n", fname, ch, lineno);
  ```

  where `fname` is the name of the file, `ch` is the invalid character, and `lineno` is the line number of the bad line.

To turn in: Put this program into a file called *life3.c*; turn it in as directed below. If you want to submit two files, that is, have this in a separate file, call the main one *life3.c* and the other file(s) *init3n.c*, where *n* is a single digit (for example, *init30.c*). Turn it in to Gradescope; you can do this as many times as you like, up to the due date, and we will grade the last one you submit.

Extra Credit

5. (*30 points*) For the program you wrote in problem 3, add the following four command-line options:
   - `-n` followed by an integer (with or without intervening space): number of generations to run the program for
   - `-p` followed by an integer (with or without intervening space): print the board at intervals of this many

   The integers must be positive. For example, running

   ```
   life3ex -n 100 -p 10 board
   ```

   would begin with the pattern in *board*, run it for 100 generations, printing every 10th one (that is, it would print generations 0, 10, 20, ..., 10, where generation 0 is the initial configuration). Always print the initial and final configurations.

   In addition to everything for the program in *life3.c*, if either option is not followed by a number, use an *`fprintf`* of the following form:

   ```
   fprintf(stderr, "%s: %c must be followed by a positive number\n", fname, option);
   ```

   where `fname` is the name of the file and `option` the option involved, and if a bad option is given, use an *`fprintf`* of the following form:

   ```
   fprintf(stderr, "%s: %c: invalid option\n", fname, option);
   ```

where `fname` and `option` are as above.

To turn in: Put this program into a file called *life3ex.c* and submit it to Gradescope.