CSI 402 – Spring 2012

Programming Assignment II

Administrative Information

- **Deadline:** 11 PM, Monday, Feb. 27, 2012.
  Cutoff: 11 PM, Wednesday, Feb. 29, 2012.
- The program must have two or more C source files.
- All the files (C source files, header files (if any) and the `makefile`) must be submitted together using the `turnin-csi402` command.
- README file

  `~csi402/public/prog2/prog2.README`

  will be available by 10 PM on Monday, Feb. 20, 2012.
- The `README` file will contain information regarding `turnin-csi402` and additional specifications for the `makefile`.

Project Description

Goal: Print a specified sequence of bytes from an input file to `stdout` in a specified base (radix).

Note: The program is modeled after `od` (octal dump) program in Unix.

Weightage: 5%

Total Points: 100 (Correctness: 85, Str. & doc: 15).

Unix Command Line:

% p2 *infile* *start* *end* *flag*

- p2: Executable version of your program.
- *infile*: Input file (may be a text file or a binary file).

Project Description (continued)

Unix Command Line (continued):

- *start*, *end*: Byte positions in the input file.
- *flag:* Specifies the radix; it may be one of `-b` (binary), `-q` (quarternary), `-o` (octal) or `-h` (hex).
- Recall that byte positions start at zero.

Example 1:

```
p2  in1.dat  230  +257  -h
```

Form of output: 10 bytes on each line, except possibly the last.

```
230:  AF  B1  C2  09  00  C9  AB  C1  D0  41
240:  AB  CD  C9  01  01  C7  AC  DC  DB  30
250:  CC  AA  BB  AC  90  81  A9  B5
```

Note: Output must be written to `stdout`.

Project Description (continued)

Example 2:

```
p2  in2.dat  174  -1  -q
```

Note: The value `-1` above indicates that all the bytes from position 174 through the last must be printed (in base 4).

Example 3:

```
p2  in3.dat  0  -1  -b
```

Note: The above command must print **all** the bytes of the specified file (in binary).

Important:

Read the specifications given on page 2 of the handout carefully.

Project Description (continued)

Errors to be detected: (Fatal errors)

- Wrong number of arguments on the Unix command line.
- Unable to open the input file.
- Invalid flag.
- Errors in *start* or *end* values. (See pages 2 and 3 of the handout for details.)

Structural Requirements:

- Your program must have **at least two C source files**.
  - One source file must contain just the `main` function.
  - One or more source files must contain functions that convert a given byte into its representation in the required base (binary, quarternary, etc.).

Additional Notes

- Read the handout carefully to understand the specifications regarding the command line, output requirements and the errors to be detected.
- Bear in mind that command line arguments *start* and *end* are strings.
- Read about `strtod` and `strtol` functions (in `<stdlib.h>`) to understand how strings of decimal digits can be converted into `int` and `long` values.
- You can use `fseek` to reach any byte position within a file.

Additional Notes (continued)

- Use appropriate bitwise operations (shifting and masking) to convert the value stored in a byte into its representation in the specified base.
- Be sure to use

  ```
  fflush(stdout);
  ```

  after each call to the `printf` function.
- Remember that your submission must contain all your C source files, header files (if any) and the `makefile`.