Assignment 1: Reading Data from Portable Network Graphics Files

Due date: *Wednesday, December 20, 2023 at 22:00*

> *This is an individual assignment. You may discuss it with others, but your code and documentation must be written on your own.*

In a source file called `pngq.c` write a C program that reads and extracts information from a Portable Network Graphics file (PNG). Use the W3C documentation of the PNG format as a reference specification.[1] The general usage of the `pngq` program is as follows:

`pngq` [*options*] [`--`] *file1* [[*options*] *file2* ...]

This means that the program takes, on the command line, zero or more *options* and one or more file names. An optional "`--`" argument (without quotes) indicates that all subsequent arguments are to be interpreted as file names. For each file name, the program must read and validate the file. That is, the program must reject an invalid PNG file with a specific error message printed on the standard error. A PNG is considered invalid if it does not conform with the specification, and also if any of the CRC check-sums does not check.

When the file is valid, the output is determined by three format strings as explained below. These are the PNG-file format or *pformat*, the chunk format or *cformat*, and the keywords format or *kformat.* They can be set with a command-line argument `p`=*pformat*, `c`=*cformat*, or `k`=*kformat*, respectively.

For all these command-line arguments, the format string is the string that directly follows the "`p=`", "`c=`", or "`k=`" prefix of this command-line argument. Multiple such arguments may appear on the command line. Each one sets the corresponding format for the files that follow on the command line. The initial, default formats are shown in Table 1.

pformat	"`_f: _w x _h, _c, _d bits per sample, _N chunks\n_C`"
cformat	"`\t_n: _t (_l)\n`"
kformat	"`\t_k: _t\n`"

Table 1: Default format strings

Output Format

For each file f, the program must output information following the currently set *pformat* string. In the format string, every character must be copied onto the output verbatim, except for the underscore character, which introduces place-holders for information on the file. The information must be derived from the chunks that make up the PNG file, including in particular the *IHDR* chunk.[2] The following placeholders are defined for a *pformat* string:

- `_f` print the file name as passed on the command line
- `_w` print the width of the PNG image
- `_h` print the height of the PNG image
- `_d` print the bit depth of the PNG image
- `_c` print the color type class as a string. The values are "Greyscale", "Truecolor", "Indexed", "Greyscale+alpha", and "Truecolor+alpha".

[1] https://www.w3.org/TR/2023/CR-png-3-20230921/
[2] https://www.w3.org/TR/2023/CR-png-3-20230921/#11IHDR

- `_N` print the number of chunk the image consists of.
- `_C` print the chunks that make up the PNG image file, each formatted according to the *cformat*
- `_K` print the keywords and corresponding text, as specified in *tEXt* chunks, formatted according to the *kformat* string.

The following placeholders are defined for a *cformat* string:

- `_n` chunk sequence number (starting from 1)
- `_t` chunk type, four-character type identifier (e.g., "IDAT")
- `_l` length of the chunk (in bytes)
- `_c` CRC32 checksum for the chunk
- `_D` chunk data. Print each byte of the content of the chunk as an hexadecimal two-digit number. Byte values on the same line are separated by a single space. A line consists of at most 16 hexadecimal values (bytes).

The following placeholders are defined for a *kformat* string:

- `_k` keyword
- `_t` corresponding text

In all format strings, any character following an underscore other than those specified for the place-holders is printed verbatim on the output. So, for example, `__` print a single underscore character ("_").

Examples

With a single file *1.png* as command-line argument, the output should be something like the following:

```
1.png: 3383 x 1949, Truecolor, 8 bits per sample, 36 chunks
1: IHDR (13)
2: pHYs (9)
3: IDAT (8192)
4: IDAT (8192)
5: IDAT (8192)
6: IDAT (8192)
7: IDAT (8192)
8: IDAT (8192)
9: IDAT (8192)

...

35: IDAT (2784)
36: IEND (0)
```

The following command processes the same single file *1.png* with the given *cformat* option. Notice that new-line characters are given directly in the single-quoted argument.

```
$ ./pngq c='_n: _t (_l)
_D
' 1.png
```

In this case, the `_D` placeholder prints the data content of each chunk. The output is as follows:

```
1.png: 3383 x 1949, Truecolor, 8 bits per sample, 36 chunks
1: IHDR (13)
 0  0  d 37  0  0  7 9d  8  2  0  0  0
2: pHYs (9)
 0  0  e c4  0  0  e c4  1
3: IDAT (8192)
78 5e ec dd 7f 74 14 f7 7d ef ff 77 62 4b a3 20
6b 24 87 1d b9 91 56 58 f2 62 49 ec 1a 7d 59 62
b0 88 64 2f d0 cb 8a 24 16 86 7e 45 ec 46 f4 ab
46 71 68 94 13 72 39 df d2 e2 94 ef c5  d 3d f8
9a 5e dc c3 a9 72 4a 4b 1d f5 aa 45 8e 5d eb d6
d8 4a 1d bc 4e 8d 65 8b b0 b6 29 eb  a 6b 91 88
d7 52 61 25 37 da 55 40 a3  8 6b 56 4e fb fd 63
57 d2 ee 68 f5  b 10 c6 f0 7c 1c 9f 63 f1 f9 cc
cc ce 8c  4 e7 cc 5b af 79 7f 3e 73 f7 dd 85 69
69 8a  0 98 c9 c8 88 21 22 86 31 62 9e  0 f0 e9
```

...

Submission Instructions

Submit one source file, as required, through the iCorsi system. Add comments to your code to explain sections of the code that might not be clear. You must also add comments at the beginning of the source file to properly acknowledge any and all external sources of information you may have used, including code, suggestions, and comments from other students. If your implementation has limitations and errors you are aware of (and were unable to fix), then list those as well in the initial comments.

You may use an integrated development environment (IDE) of your choice. However, *do not submit any IDE-specific file,* such as project description files. Also, *make absolutely sure that the file you submit can be compiled and tested with a simple invocation of the standard C compiler.*