Floating Point

Problem Description

Computers use floating-point representation to store real numbers (such as 3.14159. . . , 1.06 × 1037, -2.5) as fixed-length sequences of bits. The way that these bits are interpreted is known as the computer's floating-point format. Choosing a format involves trading off storage needed per number against the precision with which numbers can be represented, so a computer may support more than one format. Most modern computers use the two basic formats defined in IEEE Standard 754-1985[1], which are known as IEEE single-precision and IEEE double-precision respectively.

However, other floating-point formats have been in use since at least the early 1950s. One important set of formats is that introduced in 1964 with IBM's System/360 family of mainframes and used in the mainframe world ever since. System/360 also supports single- and double-precision formats, but naturally these are incompatible with the much later IEEE standard.

Task

You have been contracted to help a client migrate their IT operations from an IBM mainframe to a new desktop-based system. They have a large amount of numeric data on the mainframe which needs to be converted into a format the desktop boxes will understand. Your task is to produce a robust, carefully tested program which will read a file of IBM System/360-format floating point numbers and write them to a new file in IEEE standard format. Links (indicating the relevant page numbers) to the documents which define the formats can be found on the cosc326 website[2].

Your program should prompt for input and output filenames, and the precision (single or double) of each file. For example, if I had a filename with single precision IBM floats named *sibm.bin* and wanted an converted output in IEEE double format written to file *dieee.bin* then the program prompts and my input would look like:

```
$ ./<your program>
Enter IBM input file: sibm.bin
Specify IBM precision: s
```

[1] The IEEE Standard 754-1985 has been superseded by IEEE Standard 754-2008, but for this exercise we will stick with the 1985 version.

[2] The access to the IEEE standard requires a subscription, but University has it, and so if you follow the link from within the campus network you should get the document without issues.

```
Enter IEEE output file: dieee.bin
Specify IEEE precision: d
```

Alternately, your program can accept command line arguments in the following format:

```
./<your program> <fname1> <s|d> <fname2> <s|d>
```

Therefore, for the same example as above, your program would be run like so:

```
./<your program> sibm.bin s dieee.bin d
```

Please indicate in your submission which format your program is using.

Note that the two precision specifications are independent, so you will have to consider all four combinations. The input files will be just a binary files containing series of IBM format float numbers (in either single or double precision). The output file should be a binary file with a series of converted IEEE format float numbers (in either single or double precision).

You may need to talk to the client about how to handle exceptional cases. For the purpose of this paper the client is someone you can reach by e-mail at mark326. The inquiry about exceptional cases is considered a part of your submission, and so I would typically expect first an e-mail asking for clarification about the identified exceptional cases, which I will reply to with clarification, followed (some time later) by an e-mail with your program. Each group should independently seek the clarification on the exceptional cases – identification of the cases is considered a part of the exercise and is the reason why this transcript doesn't cover them.

Relates to Objectives

1.2, 1.4, 2.2, 2.5, 2.7, 2.8, 3.1, 3.5, 4.3, 4.4

(Group 2)