CSC236 Program Documentation Guidelines

1.0 The general level of documentation must be good enough so that:

1.1 A user of the program can prepare input data for the program and understand the output of the program.

1.2 A programmer can understand the operation of the program well enough to fix a bug or add a new function.

2.0 Specific requirements include:

2.1 A program header with these items:

To improve readability there should be a blank line between these major header sections.

2.1.1 A description of the program's function with enough detail to allow someone to use the program. It must describe: what functions the program performs; any non-standard algorithms used; what the input looks like in enough detail for a user to create input data; what the output looks like in enough detail for a user to understand the output; what errors are handled; any restrictions.

2.1.2 How the user activates or runs the program. How is data read and written from the standard input or output streams. Can one redirect input and output to files.

2.1.3 Does the program generate any prompts or error messages.

2.1.4 Does the program return a return code.

2.1.5 What are the limitations of the program.

2.1.6 The owner/programmer's name.

2.1.7 The date created.

2.1.8 The history of changes after the program has shipped. Running the grading program is equivalent to shipping the code to your customer. Once you have run the grading program, then any changes must be described in the header under the Date/Reason.

2.2 All lines of code must have a useful line comment.
*** Make sure that at least one space separates the comment from any parameter that precedes it. ***

2.3 The program must be broken into small functional blocks of code. Each block must have a header that describes the functions performed by that block of code. It should be descriptive and specific as opposed to vague and general.

2.4 Subroutines must have a header that describes the functions of the subroutine and the input and output parameters. A user must be able to prepare data for the subroutine from the header information.

2.5 Any non-standard algorithms must be explained. How does it work. Did you develop it yourself or get it from a technical reference.

3.0 Grading guidelines

Documentation is graded by software using a heuristic set of algorithms. You should document your code as well as would be expected for a real business program and then you should receive full credit.

Start your program early enough so that if you do lose documentation points, you will have the time to fix the documentation before the program's due date.

Some general guidelines to receive full credit are:
- A program header should be at least 1/4 of a typed page.
- A header block should be about 1/2 the size of the code it describes.
- At least 90% of all instructions should have line comments.
- Make sure at least one space is used to separate the four major parameters of a line of assembler code.
  ```
  label:   opcode   operands   ;line comment
  ```
- Do not use continuation lines.

4.0 Do *NOT* create "specious" documentation

Specious documentation is where the line comment just echos the opcode without providing any useful information. This is an example of specious documentation.

```
mov  ax,0      ;move zero to ax
add  ax,1      ;add 1 to ax
cmp  ax,5      ;compare ax to 5
add  bl,cl     ;bl=bl+cl
```

The automated grading system will not detect specious documentation. However, if it is detected by the staff then the instructor may manually re-grade the documentation for a submitted program ... see the syllabus for more detail.