Programming Assignment Grading Information

Your programming assignments will be graded using the following guidelines. They are pretty detailed, and may seem somewhat involved at this time, but you will get a good feel for what I am looking for as we go along. **Each programming assignments must be submitted via the appropriate Assignment link in Blackboard as an *attachment*.**

Below (next page) is a list of programming "violations" (oh my!) and point deductions for each programming violation that you might make in your assignments. I am not trying to stifle your creativity, but to introduce you to programming styles that seem to be most prevalent. Many of these concepts will become apparent as we proceed through the course. Don't worry if you don't understand some of these at this time, just keep reviewing this as we go on.

I suggest you print the following chart and keep it close to you while you develop your programs.

Continue to next page for Programming Assignment Point Deductions table...

Programming Assignment Point Deductions:

The following is a list of programming violations and the number of points which will be deducted from programming assignment grade for each violation.

Category	Description of violation	# points deducted
Comments	No block comment at beginning of program	-20
	Your Name, etc. not in program (in beginning block)	-10
	Vague, non-descriptive comments	-2 for each
	No block comments in code	-10
	Use of // 'line comments' (see notes)	-1 for each
Standards	Non-descriptive variable names*	-3 for each (*except for loop control variables)
	Variables declared, but not used	-3 for each
	Unnecessary initialization of variables	-3 for each
	Global Variables - DO NOT USE	-10 for each
	goto statements - DO NOT USE	-10 for each
	Inefficient code	-5 for each occurrence
Indentation	Not following indentation standards	-2 for each occurrence
	Inconsistent indentation	-3 for each occurrence
	Spacing - too much or too little	-3 for each occurrence
	Block Marker alignment (i.e. same indent for begin/end markers { and })	-3 for each occurrence
Correctness	Compiler Error: Program does not compile	up to -50 points
	Logic Error: Program does not produce correct results	up to -25 points
	Runtime Error: Program crashes during execution	up to -25 points
	Program output not formatted as assigned	up to -10 points for each
	Incomplete assignment	up to -50 points
Miscellaneous	Program does not conform to ANSI Standards	-10
	Program uses statements/concepts not yet introduced	-5 to -10 for each
	Lateness	-2 for each day late up to cutoff date, then no credit.

Continue to next page for a brief description of Proper Programming Standards.

Proper Programming Standards:

The following describes some good programming standards to improve the readability of a program (and to avoid miscellaneous deductions!):

Indentation - ***Always*** indent the body of a block by 2 to 10 spaces (or one tab stop). A block begins with a { and ends with a }.

Examples of these include:

- the body of a function (like function **main** for example)
- the body of an **if** statement
- the body of an **else** statement
- the body of a loop (for, while, do).

These "bodies" should be indented even if there is only a single line in the body. Single line bodies of the if, else, and loops do not require the { and }, however, they still must be indented.

Example of a 2 line body within a while loop:

```
i = 1;
while (i < 10)
{
        i = i + 1;
        printf ("Hello World\n");
} /* end while */
```

Example of a single line body with indentation:

```
for (i=1; i < 10; i++)
      printf ("Hello World\n");
```

Block Comments - Place block comments in the following locations throughout your code:

- before user input
- before a loop
- before an if/else statement
- before calculations,
- before results are output
- after every ending block marker: '}'

For example:

```
/* this is a block comment - it can be one or more lines in length, begins with slash-star and ends with star-slash */
```

Here is an example of a block comment after an ending block marker:

```
i = 10;
while (i > 0)
{
        i = i - 2;
        printf ("Hello World\n");
} /* end while */
```

Miscellaneous

1. Put spaces between operators and variables (except for ++, --, &)

    Example of correct spacing:

    ```
    for (n = 1; n <= total; n++)
    ```

    Example of incorrect spacing:

    ```
    for(n=1;n<=total;n++)
    ```

2. Separate multiple-word variable names with underscores:

    Example of correct variable name:

    ```
    number_of_values
    ```

    Example of incorrect variable name:

    ```
    numberofvalues
    ```

3. Line up variable names during declaration:

    Example of correct alignment of variables during declaration:

    ```
    int         num, total;
    long int    number_of_values
                ^ notice all variable names start in the same column spaced to the right.
    ```

Example of incorrect alignment of variables during declaration:

```
int num, total;
long int number_of_values
```

4. Block Indenting: begin a block marker under the first letter of the statement that leads the block,

Example of correct block marker alignment:

```
for (n = 1; n <= total; n++)
{
        printf ("Hello World\n");
} /* end for loop */
```

Example of incorrect block marker alignment:

```
for (n = 1; n <= total; n++)
    {
        printf ("Hello World\n");
 } /* end for loop */
```

Please see the "Assignment Submission Checklist" on the next page. This checklist will help to ensure that you do not lose points unnecessarily.

Assignment Submission Checklist

Use this to avoid the majority of deductions that students incur over and over.

- ☐ My name is in the program.
- ☐ I have a complete **header comment** as per the sample assignment solution in the Start Here section of the course notes.
- ☐ I reviewed the **indenting** and it meets the strict **standards** put forth in the course
- ☐ I reviewed the **deductions chart,** and there are no obvious deduction violations.
- ☐ The input/output is exactly as depicted in the Sample run provided in the assignment.
- ☐ I tested my program and it gives the correct results.
- ☐ I have the statement while ((c = getchar() != '\n') && c != EOF); after every scanf statement, and the variable c is defined as a char.
- ☐ I named the source file per the directions in the **assignment dropbox.**