CS 3180 – *Comparative Languages*

Coding Standards

Program Organization

General format

Every source code file should consist of:

1. A block comment including your name, the project name, and a general description of what is in this file
2. Global constants and/or variables (if any) and comments indicating their purpose
3. Function definitions with block comments (see below)
4. Main program with block comments (see below)

Use small functions

Keep functions small. The code for a function should be limited to no more than what will fit on the screen. About 50 lines should be considered an upper bound.

Comments

Each function (including main) should be preceded by a block comment describing the function, its parameters (and the data type for each parameter), its return value (and type), side effects (such as the use of [shudder!] global variables) and any known bugs or limitations.

Use inline comments

Inline comments can be helpful in definitions but should be used sparingly in procedural code. Every major section/block of code needs an inline comment for clarification. Particularly complex or difficult to understand lines of code can also be commented for clarity.

Example of a well-documented function

```
# cardDamage: adjust the current turn damage based on the current
# card modifiers.
#    Returns:  new turn damage (integer)
#    Parameters:
#       card (string) - the current card
#       base (integer) - damage so far this turn
def cardDamage(card, base):

    # Check if the card name starts with '+x' or '-x'
    # where x is a number.  If so, convert the +x or -x:
    # value to an integer and add it to the base damage
    res = re.search(r"^([\+\-][0-9]+).*$", card)

    # If the card matches the RegExp, then the bonus
    # value is in the first group (res.group(1)):
    if res:
        bonus = res.group(1)
        bonus = int(bonus)
        return base + bonus

    # Check if the card is x2 or Null:
    elif (card == "x2"):
        return base * 2
    elif (card == "Null"):
        return 0

    # everything else: don't change the base
    else:
        return base
```

Indentation and spacing

Indent consistently using a reasonable style.

Each control construct (function definition, loop, if, switch...) should define a new indentation level.

Don’t be chintzy with indentation and whitespace

- Use at least 3 spaces for each indentation level.
- Use whitespace to improve readability. Use one or more blank lines to separate major sections of code.
- Use white space to separate variables, operators, and other syntactic units.

Example:

```
x=2*x+y; <-- hard to read
x = 2 * x + y; <-- easier to read
```

Don't allow lines to grow too long

Lines longer than 80 characters or so should be split into separate lines.

Naming Conventions

Use readable English names

Constants, variables and functions should be given names that are good meaningful English. Names like I, J and Z should not be used, except occasionally as loop indexes. When names (especially function names) are stated in the program assignment handout, they MUST be used exactly as given.

multiWordVariableNames

Use either camelCase or underscores for multi-word variable names like maximumWordLength and last_name. Don't mix them. Choose one format and stick with it.

Use abbreviations sparingly

A few common abbreviations (like Num for Number) are common enough that they won't be misunderstood. It is a good idea to comment variables so their meanings are clear. If a variable is redefined, an explanatory comment is essential.

General

At this level, all programs should compile/interpret correctly, and have been well tested. There should be no infinite loops, system locking, or output that doesn't make sense. It is your responsibility to seek help for syntax and logic errors or unreasonable output. The programs should run according to the handout requirements. If there is a problem with any of these, file header comments should be included that contains all pertinent details of the problem, what steps were taken to correct the problem and any ideas you have as to what is wrong.