Pseudocode & Flowchart Guidelines

Adapted from John Dalbey, Bob Roggio and Farshad Barahimi.

Pseudocode and flowcharts allow the designer to focus on the logic of the algorithm without being distracted by details of language syntax. At the same time, the pseudocode or flowchart needs to be complete. It describes the entire logic of the algorithm so that implementation becomes a matter of translating from the pseudocode or flowchart into source code.

Pseudocode

In general the vocabulary used in the pseudocode should be the vocabulary of the problem domain, not of the implementation domain. The pseudocode is a narrative for someone who knows the requirements (problem domain) and is trying to learn how the solution can be organized. That is, someone who is a non-programmer should be able to read and understand your pseudocode. For example:

Extract the next word from the line (good)
Set word to get next token (poor)

Append the file extension to the name (good)
Name = name + extension (poor)

For all the characters in the name (good)
For character = first to last (poor)

Note that the logic must be decomposed to the level of a single loop or decision. Thus "Search the list and find the customer with highest balance" is too vague because it takes a loop **and** a nested decision to implement it.

Each textbook and each individual designer may have their own personal style of pseudocode. Pseudocode is not a rigorous notation, since it is read by other people, not by the computer, but **this is the style you should follow for this class.**

Examples:

```
Get student's grade from user
If student's grade >= 60
    Print "passed"
else
    Print "failed"
```

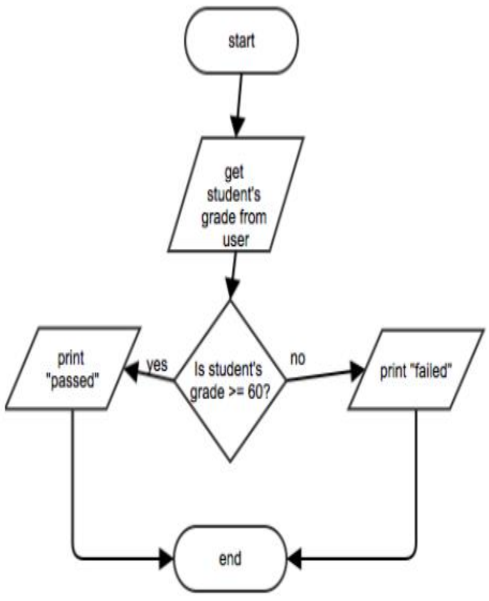
```
Set total to 0
Set grade counter to 1
While grade counter <= 10
    Input the next grade
    Add the grade to the total
    Add 1 to the grade counter
Set the class average to the total divided by 10
Print the class average
```

Flowcharts

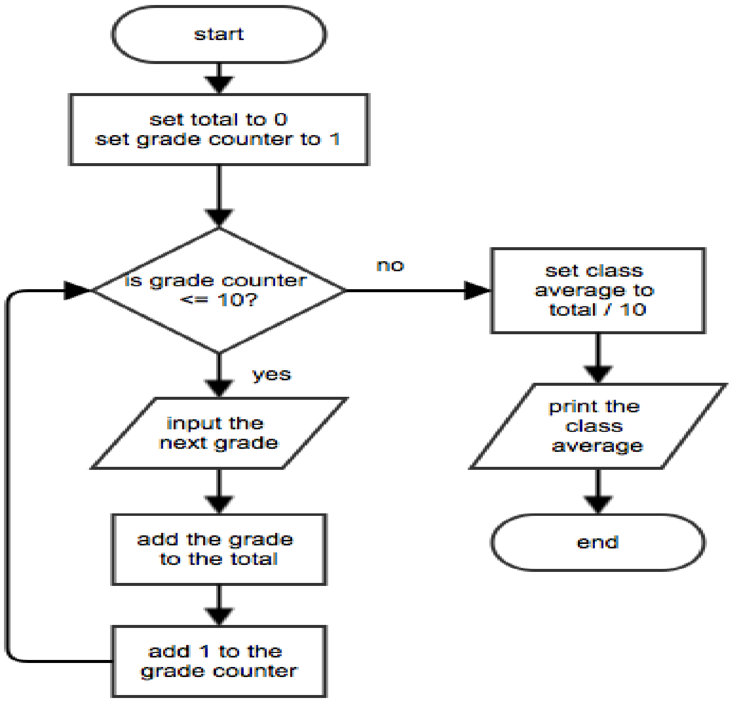
A flowchart depicts an algorithm in a graphical format. Here are descriptions of the most common symbols:

- Oval: start and end
- Arrow: represents flow of control in a program.
- Rectangles: shows a computation or a specific process, for example "multiply X by 2".
- Parallelogram: Used for getting **input** from user or sending **output** to user.
- Diamond: Used for conditional flow control where a program has to decide which way to go, for example: if X is divisible by 2 do this thing; if not, do this other thing. Without conditional flow control, a program would have just one path from start to end, but with it there may be many different paths from start to end.

Examples:



start
get student's grade from user
Is student's grade >= 60?
yes
no
print "passed"
print "failed"
end



start
set total to 0
set grade counter to 1
is grade counter <= 10?
no
yes
input the next grade
add the grade to the total
add 1 to the grade counter
set class average to total / 10
print the class average
end