

APM 153 LECTURE SEVEN Pseudocode and Flowchart for `packageweight.m`,
Formatted Output, Notes about Schedule.

Algorithm for `packageweight.m`

- (1) The program for Assignment Three should be named **`packageweight.m`**.
- (2) For Assignment Three you will need to finish your program and write out your algorithm as pseudocode and as a flowchart.
- (3) As a rule, you should write your **algorithm first**, and **then** write your program!
- (4) Listed below are some of the steps we need to follow for our `packageweight` algorithm. Just like the jumbled brownie recipe, what order should they be in?

- A. write out cost using `fprintf` _____
- B. If package more than 100 pounds write out “Sorry, package too heavy” _____
- C. round weight of package up using **`ceil`** function _____
- D. If weight less than or equal to 2 pounds, cost equal to \$10 _____
- E. Prompt user for weight of package. _____

(5) For `packageweight.m` to work properly, our algorithm needs an extended if-then statement, like the one in `calc_roots.m` that went something like the lines of code below.

```
if      blah blah blah
        do this

elseif blah-biddy blah blah
        do this other thing

else
        do this last thing

end
```

(6) At each step in the extended if-then statement, the algorithm deals with the cost of packages in different weight classes. For example,

if weight is less than or equal to 2

cost = _____
display cost with message

elseif weight is greater than two and weight is less than or equal to 70

cost = _____
display cost with message

elseif weight is greater than 70 and weight is less than or equal to 100

cost = _____
display cost with message

else

display message that package is too heavy.

end

(7) For each of the three blank lines above, what should be the cost?

(8) On pages 105 and 106, your textbook lists the algorithm for calc_roots.m should you need an example of what pseudocode should look like.

(9) One thing you might notice is how the lines of code written inside of if-then statements are **indented**.

(10) Even in the Matlab GUI, when we type in an if-then statement, it automatically indents then next line of code.

(11) Indenting the lines of code inside each part of an if-then statement helps us **keep track** of what is going on inside each part. This is especially useful when we are writing long, extended if-then statements with lots of **ifs and elseifs**.

(12) Don't forget the **end** at the end of each if-then construct and **elseif** is **one word**.

(13) Use this page to write out the flowchart for `packageweight.m`

Formatted Output Using fprintf

(14) To control how Matlab writes output, you need to use the **fprintf** command. In the `calc_roots.m` script file, the output is written using the format **%f\n** as in the line

```
fprintf('x1 = %f\n', x1)
```

(15) The first part **'x1 =** prints out the character string **x1 =**

(16) The next part, **%f\n**, are formatting symbols that tell Matlab to **how** to print what's next..

(17) The **%** is the formatting symbol in Matlab and the symbols following the **%** are the formatting instructions.

(18) Table 4.2 on page 42 of your textbook summarizes the most common characters used in `fprintf` formats.

%f - displays value as a floating point number

%d - displays value as an integer

%e - displays value using exponential notation

\n - acts as "hard return"

(19) We use `fprintf` when we want to print out more than one thing on a line of output.

(20) The `disp` command can only print out one thing on each line.

Notes About Our Schedule

(21) Only Five More Weeks till Spring Break! – Remember, DO NOT plan to blow off class!

(22) Our first exam will be on Friday, February 17th in lab during our regular class period.