# PROG 1700 - Assignment 3

# DECISIONS AND FUNCTIONS

Assignment Value: *8*% of overall course mark.

Due Date: **See due date designated on the Assignment 3 dropbox on D2L.**

Late submissions will receive the standard late submission penalty as stated in the course outline. (5% overall deduction per day late, until 60%, and 0% after assignment handed back to the class.)

#### Assignment Instructions:

Use PyCharm to create console applications (.py files) in which you’ll code the answer for each of the following problems. **You must create a new .py file for each question in this assignment.**

#### Submissions:

Once you’ve completed all three required programs for the assignment, add all three .py files to a single .ZIP file. The .ZIP file name should include your name, W#, section # and Assignment 3 (ex. DoeJohn\_w0123456\_702\_Assignment3.zip).

Upload the .ZIP file to the D2L dropbox labelled Assignment 3, which can be found under the Assessments Dropboxes link.

#### Evaluation:

To insure the greatest chance of success on this assignment, be sure to check the marking rubric contained at the end of this document or in D2L. The rubric contains the criteria your instructor will be assessing when marking your assignment.

## Program 1 – Desk Calculator

A carpenter needs a program that computes the price of any desk a customer orders based on the following: order number, desk length in inches and width in inches, type of wood (“mahogany”, “oak” or “pine”), and number of drawers. The price computed as follows:

• The charge for all desks is a minimum $200.

• If the surface (length \* width) is over 750 square inches, add $50.

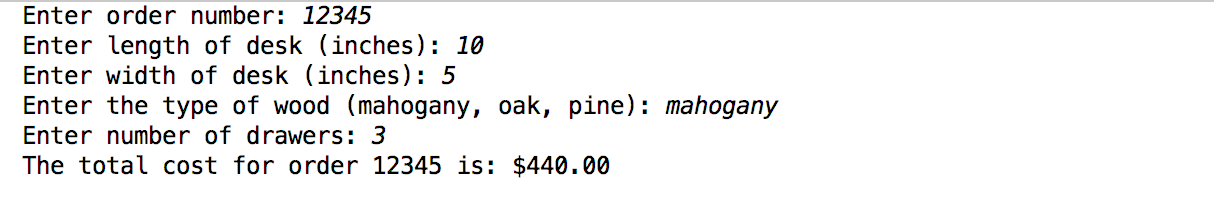
• If the wood is “mahogany” add $150; for “oak” add $125. No charge is added for “pine”.

• For every drawer in the desk, there is an additional $30 charge.

First, create a flowchart that clearly shows all of the paths of execution that will exist within your designed solution to this problem. Then, write a console application that will input the order number, desk length, desk width, type of wood and number of drawers and then output the corresponding price.

Examples and Testing

In the section below you will be presented with at least one screenshot of a successful execution of a sample solution to the program, which should help demonstrate how your input/output on the program should work. In addition to the sample values used in the screenshot(s), additional testing values are given in a chart along with the output values that they should produce. You can expect your instructor to grade your assignment by using all of these listed input values at the very least, but keep in mind that additional values may also be used as well. In other words, **you should thoroughly test your code before submitting!**

**Sample Output** - Make sure your program can output data *exactly* as shown below.  


**Other testing values**;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Order Number | Length | Width | Type | No. of Drawers | Total Cost |
| 55555 | 150 | 30 | oak | 4 | $495.00 |
| 11111 | 48 | 36 | pine | 2 | $310.00 |

## Program 2 – Eureka Water Company

Eureka Water Company charges the homeowner a single rate based on the total amount of water used by a customer in the billing period. The amount charged will be calculated based on which of the following ranges their total water usage falls under. Note, it is not a cumulative charge. Your program will figure out which single range the usage falls into, then calculate the cost based on that range’s cost.

|  |  |
| --- | --- |
| Total Water Usage | Rate of Charge |
| Up to and including 1000 cubic feet | $15.00 flat rate |
| Over 1,000 cubic feet and up to and including 2,000 cubic feet | $0.0175 per cubic foot |
| Over 2,000 cubic feet and up to and including 3,000 cubic feet | $0.02 per cubic foot |
| Over 3,000 cubic feet | $70.00 flat rate |

First, create a flowchart that clearly shows all of the paths of execution that will exist within your designed solution to this problem. Then, write a console application to input the customer’s usage in cubic feet and output the corresponding rate charge.

Examples and Testing

In the section below you will be presented with at least one screenshot of a successful execution of a sample solution to the program, which should help demonstrate how your input/output on the program should work. In addition to the sample values used in the screenshot(s), additional testing values are given in a chart along with the output values that they should produce. You can expect your instructor to grade your assignment by using all of these listed input values at the very least, but keep in mind that additional values may also be used as well. In other words, **you should thoroughly test your code before submitting!**

**Sample Output** - Make sure your program can output data *exactly* as shown below.

**Other testing values**;

|  |  |
| --- | --- |
| Cubic Feet of Water | Total Charge |
| 875 | $15.00 |
| 1922 | $33.64 |
| 3412 | $70.00 |

## Program 3 – Tax Schedule

You will write a program that computes taxes according to the following schedule:

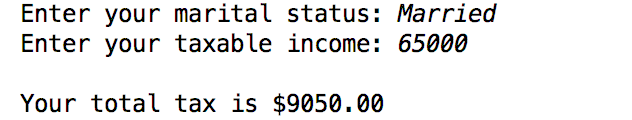
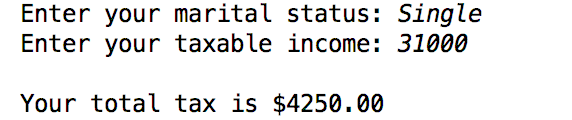
|  |  |  |  |
| --- | --- | --- | --- |
| **If your status is ‘Single’ and if the taxable income is over** | **but not over** | **the tax is** | **of the amount over** |
| $0 | $8,000 | 10% | $0 |
| $8,000 | $32,000 | $800 + 15% | $8,000 |
| $32,000 |  | $4,400 + 25% | $32,000 |
| **If your status is ‘Married’ and if the taxable income is over** | **but not over** | **the tax is** | **of the amount over** |
| $0 | $16,000 | 10% | $0 |
| $16,000 | $64,000 | $1,600 + 15% | $16,000 |
| $64,000 |  | $8,800 + 25% | $64,000 |

First, create a flowchart that clearly shows all of the paths of execution that will exist within your designed solution to this problem. Then, write a console application that will input the marital status and taxable income and then output the corresponding total tax.

Examples and Testing

In the section below you will be presented with at least one screenshot of a successful execution of a sample solution to the program, which should help demonstrate how your input/output on the program should work. In addition to the sample values used in the screenshot(s), additional testing values are given in a chart along with the output values that they should produce. You can expect your instructor to grade your assignment by using all of these listed input values at the very least, but keep in mind that additional values may also be used as well. In other words, **you should thoroughly test your code before submitting!**

**Sample Output** - Make sure your program can output data *exactly* as shown below.

**Other testing values**;

|  |  |  |
| --- | --- | --- |
| Marital Status | Taxable Income | Total Tax |
| Married | 42000 | $5500.00 |
| Married | 7500 | $750.00 |
| Single | 6599 | $659.90 |
| Single | 45000 | $7650.00 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Program 1 – Desk Calculator** | |  |  |  |  |  |
| **Criteria** | **Unsatisfactory (0 pts)** | **Partially Correct (1 pt)** | **Excellent (2 pts)** | **Marks** | **X** |  |
| **Flowchart** | Little to no effort was made to plan out the program using a flowchart that demonstrates the necessary paths through the program based on all potential decision points. | A reasonable but incomplete effort was made to plan out the program using a flowchart that demonstrates the necessary paths through the program based on all potential decision points. | A comprehensive effort was made to plan out the program using a flowchart that demonstrates the necessary paths through the program based on all potential decision points. |  |  |  |
| **Input / Output** | Little or none of the elements in the Excellent column are present | Most of the elements in the Excellent column are present | All of:   * The five user input values can be successfully captured through descriptive prompts |  |  |  |
|  |  |  | * The single output line is well-formatted and contains all expected information * Output amounts displayed using proper currency formatting (e.g. preceded by a $ symbol, two decimal places) |  |  |  |
| **Variables & Data Types** | Solution displays little to no understanding of variable usage.  Data rarely used or stored in correct data type. | Solution displays some understanding of variable usage.  Most data is stored using correct data types or some errors in casting exist | Solution displays strong understanding of variable usage.  All data is stored using correct data types and cast to other data types when required. |  |  |  |
| **Wood Type** | Wood Type not determined from user input | Wood Type sometimes determined from user input | Wood Type always determined from user input and program can handle upper and lower case entries, or any variation thereof. |  |  |  |
| **Total Cost** | Total cost amount calculation is not attempted or contains multiple errors | Total cost amount is calculated based on variable values, but there is an error | Total cost amount is correctly calculated and output to the console. |  | 2 |  |
| **Comments & Best Coding Practices**  (At least 60% of the functional requirements must be complete) | Little to no organizational or explanatory comments used  No apparent naming convention was followed or was inconsistently applied  Source code was poorly formatted | Some organizational or explanatory comments are used, some are meaningful and easily understood  A naming convention was used for part of the program, but deviated often  Effort was made to format the code, but improvements could be made | Organizational or explanatory comments are used extensively, most are meaningful and easily understood  A consistent naming convention was used for most of the program and deviated very little  Source code was clean, consistently well-formatted and easy to read |  | 2 |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | **Total:** |  | **/16** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| **Program 2 - Eureka Water Company** | |  |  |  |  |
| **Criteria** | **Unsatisfactory (0 pts)** | **Partially Correct (1 pt)** | **Excellent (2 pts)** | **Marks** | **X** |
| **Flowchart** | Little to no effort was made to plan out the program using a flowchart that demonstrates the necessary paths through the program based on all potential decision points. | A reasonable but incomplete effort was made to plan out the program using a flowchart that demonstrates the necessary paths through the program based on all potential decision points. | A comprehensive effort was made to plan out the program using a flowchart that demonstrates the necessary paths through the program based on all potential decision points. |  |  |
| **Input / Output** | Little or none of the elements in the Excellent column are present | Most of the elements in the Excellent column are present | All of:   * The single user input value can be successfully captured through a descriptive prompt * The single output line is well-formatted and contains all expected information * Output amount displays using proper currency formatting (e.g. preceded by a $ symbol, two decimal places) |  |  |
| **Variables & Data Types** | Solution displays little to no understanding of variable usage.  Data rarely used or stored in correct data type. | Solution displays some understanding of variable usage.  Most data is stored using correct data types or some errors in casting exist | Solution displays strong understanding of variable usage.  All data is stored using correct data types and cast to other data types when required. |  |  |
| **Total Charge** | Total charge amount calculation is not attempted or contains multiple errors | Total charge amount is calculated based on variable values, but there is an error | Total charge amount is correctly calculated based on variable values. |  | 2 |
| **Comments & Best Coding Practices**  (At least 60% of the functional requirements must be complete) | Little to no organizational or explanatory comments used  No apparent naming convention was followed or was inconsistently applied  Source code was poorly formatted | Some organizational or explanatory comments are used, some are meaningful and easily understood  A naming convention was used for part of the program, but deviated often  Effort was made to format the code, but improvements could be made | Organizational or explanatory comments are used extensively, most are meaningful and easily understood  A consistent naming convention was used for most of the program and deviated very little  Source code was clean, consistently well-formatted and easy to read |  | 2 |
|  |  |  | **Total:** |  | **/14** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| **Program 3 - Tax Schedule** | | |  |  |  |
| **Criteria** | **Unsatisfactory (0 pts)** | **Partially Correct (1 pt)** | **Excellent (2 pts)** | **Marks** | **X** |
| **Flowchart** | Little to no effort was made to plan out the program using a flowchart that demonstrates the necessary paths through the program based on all potential decision points. | A reasonable but incomplete effort was made to plan out the program using a flowchart that demonstrates the necessary paths through the program based on all potential decision points. | A comprehensive effort was made to plan out the program using a flowchart that demonstrates the necessary paths through the program based on all potential decision points. |  |  |
| **Input / Output** | Little or none of the elements in the Excellent column are present | At least one of the elements in the Excellent column are present | Both of:   * The two user input values can be successfully captured through descriptive prompts * The single output line contains all expected information and formatted as shown |  |  |
| **Variables & Data Types** | Solution displays little to no understanding of variable usage.  Data rarely used or stored in correct data type. | Solution displays some understanding of variable usage.  Most data is stored using correct data types or some errors in casting exist | Solution displays strong understanding of variable usage.  All data is stored using correct data types and cast to other data types when required. |  |  |
| **Single or Married** | Not implemented or contains too many errors. | Implemented, but contains at least one error, or accomplished using inappropriate methods. | Program correctly decides/branches based on marital status. |  |  |
| **Tax Calculations** | Not implemented or contains too many errors. | Implemented, but contains at least one error, or accomplished using inappropriate methods. | Final tax cost is correctly calculated based on user-entered values. |  | 3 |
| **Code Efficiency** | Code demonstrates little to no understanding of applicable concepts in developing an efficient solution. | Code demonstrates a reasonable, but incomplete, understanding of applicable concepts in developing an efficient solution. | Code demonstrates a strong understanding of applicable concepts in developing an efficient solution. |  |  |
| **Comments & Best Coding Practices**  (At least 60% of the functional requirements must be complete) | Little to no organizational or explanatory comments used  No apparent naming convention was followed or was inconsistently applied  Source code was poorly formatted | Some organizational or explanatory comments are used, some are meaningful and easily understood  A naming convention was used for part of the program, but deviated often  Effort was made to format the code, but improvements could be made | Organizational or explanatory comments are used extensively, most are meaningful and easily understood  A consistent naming convention was used for most of the program and deviated very little  Source code was clean, consistently well-formatted and easy to read |  | 2 |
|  |  |  | **Total:** |  | **/20** |
| Assignment 3 Total Mark: \_\_\_\_\_\_\_\_\_ / 50 | | | | | |