Assignment Structure for Programs

Your **comments** at the top of your work/flow chart or program, should contain the following:

* The Unit (example Unit 2), the Assignment number (example Assignment 1) and the Question number.
* Your Name and date you began creation/coding the question. (at the top for coding, wherever clear on flowcharts)
* Modifications you made, dated (for coding - last pieces at the top before the program begins.)
* The question or problem you were asked to solve/produce/create - at the top for coding.
* A MUST, all work, flowcharts, the Code pieces you borrowed, and ideas must be properly acknowledged (Comments in your code, footnotes on other pieces). You need to state, from where you borrowed, book/page/teacher/class notes with date/friend etc. and the link (You are encouraged to borrow “good” code and acknowledge the author/creator/writer). We borrow code all the time in the field, and we acknowledge the author. Failure to properly acknowledge what is borrowed is plagiarism and earns a zero on Communications and possibly the entire question. NOTE: You may not borrow more than 35% of your code.
* White space to make all of this easy to read
* Any notes or cautions about quirky things or test data.

**Almost every line of code should have a comment.** You can block comment. {You may use a comment for 2 or 3 rows, like “ setting up variables and constants with zero’s to eliminate infinite loops”.



**IMPORTANT - Program Naming**: Please name your Assignment programs/folders by FirstNameLastInitial, followed by U1A1Q# (for Unit 1, Assignment 1, Question \_\_). The folder containing everything, does not have questions numbers. For example; Mohammed’s version of Question 4 on the assignment would be ***MohammedAU1A1Q4.***

**Note:** If your first name, last initial is not unique, you will need another initial on your last name until you are uniquely identified. (ie: Mohammed, Dan, Kevin, Adam, Sean etc.)

Your **Assignment needs to be compressed** (zipped for Windows) **and uploaded** to Google Classroom (not Linked) – please ask if you have never uploaded a file or compressed a group of files. Remember to click the “Turn In” twice or I don’t have it for marking.

For example: The zipped file (MohammedAU1A1) Assignment 1 would contain;

1- A Folder MohammedAU1A1 (and all subfolders {java files, any flowcharts and text documentation.(docx or other))labeled similarly MohammedAU1A1Q1Flowchart,… MohammedAU1A1Q5*program*.java

The rubric applies to each question, but the weighting changes slightly for each question.

(We may make minor modifications to the rubic tomorrow in class - make sure you have the most recent version.)

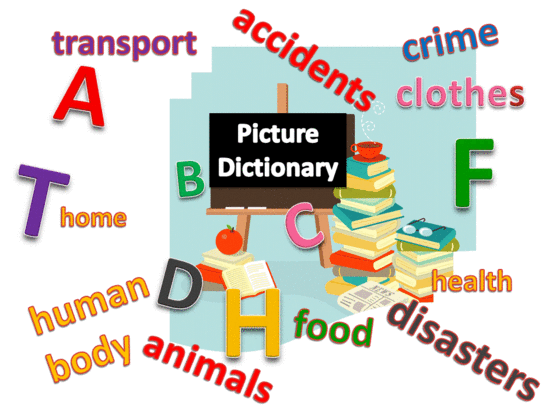
One last note: Any images or photo’s must be your originals or Common License. (No Plagiarized images.)

# **Generic Rubric for Programming Assignments – will work on in class**

Note: Some of these items will not be marked. Obviously debugging and testing cannot be marked until taught – When in doubt ASK!

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Achievement Category/Skill/  Practice (level 3, expectation | Level 4  (80-100%) | Level 3  (70-79%) | Level 2  (60-60%) | Level 1  (50-59%) | Below Level 1 |
| **Communication** | | | | | |
| **User Interface:**  ✦Spelling and grammar perfect  ✦ Clear and concise language | N/A | Perfect spelling & grammar for user | ✦Fair  ✦Clear but too concise seems abrupt | ✦Poor  ✦Uncertain or confusing | Incomplete |
| **Readability**:  ✦code is well organized and very easy to follow. (spelling and grammar ignored as long as readable and understandable on internal documentation)  ✦ Follows standard programming practices (IE: block of code well documented) or documented inline or above if comment too long.  ✦ Writers and authors acknowledged properly | Code unique and explained  efficiency Matches your program | -clear & concise  -white space  -useful comment  Neat – straight lines – arrows – y/n t/f (intent rather than code) | ✦ **some**          ✦**some** | ✦ **limited**          ✦**few** | Incomplete |
| **Application** | | | | | |
| **Use of programming structures:**  (will vary depending upon Assignment) - flowcharts/IPO/pseudo-code/Proj. Mgmt/Ghant charts etc., | >7=76 % match with final code | <= 75% or less matched perfectly | ✦Fair | ✦Poor | Incomplete |
| **Debugging**  ✦either contains statements showing effective debugging/ error avoidance or obviously done through using tools; code error free for purposes of the assignment | ‘Dumb user’ proofed  -better beyond scope or expectations | Obvious defaults work – dumb user – expected stuff handled | ✦Somewhat effective | ✦Limited effectiveness |  |
| **Test Data:**  ✦ Thoroughly tested and test data submitted | (covered off weird stuff too?) | Everything works as expected | ✦Developssome data to test | ✦Develops little or no data to test |  |
| **Solution to Problem:**  ✦ produces correct output for all or almost all data cases – (matches output sample given on problems where a standard output is expected.) | Value added for user (nice – thank you etc.) | Correct output | ✦Fair | ✦Poor |  |
| **Thinking & Inquiry** | | | | | |
| **Problem Definition:** ✦clearly states and addresses all components of the problem | -value added, play again?/something similar, an additional piece | Addressed all asked requirements | ✦Fair | ✦Poor |  |
| **Coding Efficiency:**  ✦code is efficient without sacrificing readability and understanding | Modular program(multiple calls to the same function, classes reusable, inheritance etc.) | Reusability assumed | ✦Pieced together | ✦Poor design or limited understanding evident |  |
| **Appropriate use of Structures:**  ✦ (names of variables, classes, methods…) make sense and are intuitive to a programmer | Overloading with variables/parameters | Expected as listed – required. | ✦Fair | ✦Poor |  |
| **Knowledge** | | | | | |
| **Correct usage of:**  ✦ procedures  ✦ terminology  ✦ concepts (appropriate to assignment and current lessons) | -beyond current learning (new) or unique or different usage of something you know (efficiency) | Expectations listed, correct concepts terminology – what we’ve learned | ✦Considerable | ✦Some or Fair |  |
| **Marks:** | | | | | |
|  | Communication: | Application: | Think. & Inq.: | Knowledge: |  |
| **Comments:** |  |  |  |  |  |
| **Marks Split:** |  |  |  |  |  |
| **Overall Mark:** |  | | | | |

Assignment # 1 - out of 65.

1) Make a flowchart, then write a program that checks the ratings of 5 of your favorite movies and places them in the dictionary. Give the user some instructions, then allow them to check for possible entries in your dictionary.

Application **/4** Communication **/5**

Thinking & Inquiry **/3** Knowledge  **/3**

**[ Total marks /15 ]**

2) (Basic Methods – redo – create the Swift version of this!) Craps is a popular dice game played in casinos. Create a flowchart and write a program to play a variation of the game, as follows;

Roll 2 dice. Each die has six faces representing values 1, 2, …, and 6, respectively. Check the sum of the two dice. If the sum is 2,3,, or 12 (called *craps*), you lose; if the sum is 7 or 11 (called *natural*), you win. If the sum is another value (i.e., 4,5,6,8,9, or 10.), a *point* is awarded. Continue to roll the dice until either a 7 or the same point value is rolled. If a 7 is rolled, you lose. Otherwise you win.

Your program acts a single player.

Possible samples:

You rolled 5 + 6 = 11

You win!

You rolled 1 + 2 = 3

You lose.

You rolled 4 + 4 = 8

Your points are 8

You rolled 6 + 2 = 8

You win!

You rolled 3 + 2 = 5

Your points are 5

You rolled 2 + 5 = 7

You lose.

Application (includes flowchart) **/4** Communication (includes flowchart) **/5**

Thinking & Inquiry **/3** Knowledge  **/3**

**[ Total marks /15 ]**

3) (ArrayQuestion) I’m sure many of you know of or have seen the movie “The Bucket list” with two very famous actors. You task is to create your own bucket list, but not just any bucket list. Here are the requirements;

a) It will be a **var** with type **String**

b) You should be able to append to it

c) You should be able to remove items from it (that may no longer interest you)

d) Use subscript to find your top 3 items in the bucket list – (this item may be removed depending upon how far we get in a short week.) (???)

e) Ask you teacher for their bucket list (which will change daily) and compare the two lists.

f) Create a summary of what you have done and discovered.

Application (includes flowchart) **/4** Communication (includes flowchart) **/5**

Thinking & Inquiry **/3** Knowledge  **/3**

**[ Total marks /15 ]**



4) Impress me with your knowledge of Chapter 1 and 2.

Create a flowchart and then write a program that uses any the above listed elements.

Have fun with this one!

Application (includes flowchart) **/5** Communication (includes flowchart) **/5**

Thinking & Inquiry **/5** Knowledge  **/5**

**[ Total marks /20]**