**Wayne State University**

**Upload Date: Feb 8, 2024**

**CSC 4110 - Software Engineering**

**Weekly Homework**

**Video Link:** [**https://www.youtube.com/watch?v=-t9MotIkS-k**](https://www.youtube.com/watch?v=-t9MotIkS-k)

**Directions:**

**Do all problems by the due date. Follow instructions explicitly. See general requirements for each problem.**

**There are TWO parts to this assignment:**

**CODING and QUESTIONS/ DOCUMENTATION**

**Part One: CODING**

See **General Requirements (pg 5)** - some **Problems have ‘Specific’ requirements**

**Directions:**

* Customer requests to be fulfilled **ON-TIME**
* .py as .txt file to be uploaded to GitHub repository (see below links)
* GitHub link placed in COMMENT section of Upload folder
* Code images and output images placed in THIS original assignment, uploaded to course shell (with appropriate comments, etc…)
* Comments appropriate and explanatory, contain all necessary information

**Problem One**

**\*\*\*\* Adhere to ‘General Requirements’**

**Assignment: Non-disparate DATA WAREHOUSING**

Simulate non-disparate data warehousing with the following **sequence**:

**Step One: ‘data collector’**

Create a **data collector** method, simulating user records with the following attributes: username, password, birthdate, address, social security number,productPurchased,salesperson. This program **procedurally generates ‘sample’ data and stores that data.**

**Step Two: ‘key/value’ pairs**

Feed data collector values into ***key-value pair***. For example, the user data may be an entire list sequence, which is then considered the ‘value’ portion of a unique user ID key.

**Step Three: search engine**

This key-value structure must be searchable. For example, a user may be able to search the entire data store for users in a certain state, or see which users were handled by a certain salesperson (or sales ID).

**Problem One Requirements**

“ProductPurchased” consists of order/vendor information such as usernames, web orders, product IDs, quantities, date of order, region, etc.. Any item referring to products, such as Product ID should have the prefix “ID,” such as “ID-trxdfn.”

*The specific design, method and procedure details are up to the student; the column/ category names are up to the student.*

***The student MUST create a DATA COLLECTOR method that PROCEDURALLY generated USER DATA; then the DATA must be placed into key/value pairs and be searchable.***

**ProblemTwo**

**\*\*\*\* Adhere to ‘General Requirements - LAST PAGE’**

**Assignment: Create a ‘Game of Chance’**

Create a ‘game of chance’ simulation to do the following:

(a) build and populate treasure chest with as many items customer requires

(b) create a bank / loot stash

(c) wagers to be placed per “spin” or treasure chest “grab”

(d) customer “plays” until bank account reaches 0 or below.

**Problem Two Requirements:**

Note: the name of the simulation shall be “pirate” related; copy/ paste code and output, showing different outcomes; “**random**” module is to be imported.

**ProblemThree**

**\*\*\*\* Adhere to ‘General Requirements - LAST PAGE’**

**Assignment: Create password simulator**

Customer needs a password simulator to do the following:

(a) create random passwords in perpetuity

(b) if the password is “acceptable,” it gets archived

(c) “unaccepted” passwords get deleted

(d) no less than 40 iterations

**Problem Three Requirements:**

Customer rules of ‘accepted passwords’ include: “special symbols,” and password cannot be a word in a dictionary list; “**random**” module to be imported.

**General Requirements:**

1. **Add labeling/ comments (name, date, revision #); add in-line requirements where appropriate (such as syntax usage).**

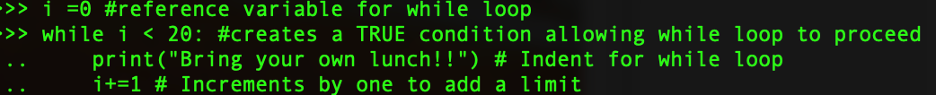
1. **AT LEAST ONE PROBLEM MUST USE SONIFICATION AND VISUALIZATION.**

#Indicate coding begin and end

Example acceptable code comment:

**# Revision number BEGIN/ START DATE**

**## Begin John D. Student here (date)**



**# Revision number FINAL DATE**

**## End John D. Student here**

**# Group / manager/ lead tech/ project # ←-Where appropriate**

1. **Adhere to the following coding style (from PEP8):**

1. Wrap lines so that they don’t exceed 79 characters.

2. Use blank lines to separate functions and classes, and larger blocks of code inside functions

3. When possible, put comments on a line of their own.

4. Where appropriate, name your classes and functions consistently; the convention is to use UpperCamelCase for classes and lowercase\_with\_underscores for functions and methods.

1. **GitHub:**

GitHub Video 1: <https://www.youtube.com/watch?v=fJtyf62yAb8>

GitHub Video 2:<https://www.youtube.com/watch?v=GqNAD4XoZ6k>

**Reference following article to create repository so you can load this assignment output:**

**https://docs.github.com/en/desktop/installing-and-configuring-github-desktop/overview/getting-started-with-github-desktop**

1. **Fill out below CRD**

**Change Request Document**

**Name:**

**Student access ID:**

**Project:**

**Date:**

**Group Number:**

***Everything in italic should be changed as appropriate by you and should not be italic when submitted****. Also remember code is not changed until the Refactoring stage, so don’t put “I changed” or similar until section 4 of the report.*

*(Title of the change request*)

**1.** **Change Request and concepts*:***

*In this section, describe the change that you were requested to do. Give any relevant background information or any essential details.*

*Extract significant concepts and list them here.*

***2.*** ***Sources:*** *Include any sources that you cited or used information from*

***3.*** ***Highlighted Source Code:***

*Attach or cut and paste the code of the classes that you changed. Highlight the code that was changed or added. Use YELLOW for modified code RED for deleted code, and GREEN for added code.*

*If you only changed one method in a large file, only include that method and the file name it’s from. Likewise, if you only changed a line or two in an event map or resource file, only include a few of the surrounding lines and the file name. Do not include thousands of lines of code that you did not change!*

**Part Two: QUESTIONS**

**Answer End of Chapter Questions with Real-Life Examples, documented by APA references, at least ONE reference per Question.**

See following link to automatically CREATE your references:

<https://owl.purdue.edu/owl/research_and_citation/apa_style/apa_formatting_and_style_guide/general_format.html>

**Question One**

What are the main differences between a ‘data warehouse’ and a typical SQL database?

**Give examples and references.**

**Question Two**

What are differences between someone tracking expenditures via an Excel spreadsheet versus an SQL database? (eg. scale)

**Give examples and references.**

**Question Three**

In compiled languages, what steps do programmers do to produce an executable file?

**Give an example.**

**Question Four**

What is the role of version control systems in software projects?

Give two examples of Github-like applications/ programs and cite their differences.

**Question Five**

What is a ‘**diff**’ file?

Paste a screenshot of a **diff** file here (showing side by side)

**Question Six**

Explain what a baseline is.

**Question Seven**

How does the program version in the private workspace differ from the baseline version?

What is a conflict in terms of two different updates to a file? How does it get resolved?

**Question Eight**

What is the build and what is the result of the build?

**Question Nine**

What is the three-tier architecture?

**Question Ten**

What is polymorphism in technology? Give an example.

NOTE:

