Question 1:

(a) Reasons for Coding Plagiarism:

* Plagiarism in computer programming occurs when one person submits another's code as his or her own.
* The abundance of online resources makes it easy to be careless with copying and pasting. Failure to understand copying code without understanding how it works, usually because they don't know enough or don't have enough time.
* Due to a lack of experience, novice programmers may not know how to reference or adapt code from other sources properly.
* For example: In COVID-19, NUS students who cheat on a take-home test receive no credit for the work they submit. (Wong Yang, 2020). According to the results of the plagiarism checks, some of the students in this class, which consists of engineering students, plagiarism during this hands-on exam.

Ways to avoid plagiarism in coding:

* Use only code that you have a firm grasp on; don't plagiarize until you know precisely what you're doing.
* Honesty and integrity should serve as guiding principles in your coding endeavors.
* Make use of existing code as a starting point but create your own answer.

(b)

Python code to run a password generator that produces secure passwords.

(Simran Kaur Arora,2 Septemeber,2023)

Code:

import secrets

import string

def create\_strong\_password(powerful\_password=12):

char = string.ascii\_lowercase + string.ascii\_uppercase

digits = ''.join(str(i) for i in range(10))

interest\_char = string.punctuation

elements = char + digits + interest\_char

exclusive\_password = ''

tough\_password = False

while not tough\_password:

exclusive\_password = ''.join(secrets.choice(elements) for \_ in range(powerful\_password))

has\_special\_chars = any(char in interest\_char for char in exclusive\_password)

has\_digit\_count = sum(char.isdigit() for char in exclusive\_password) >= 2

if has\_special\_chars and has\_digit\_count:

tough\_password = True

return exclusive\_password

if \_\_name\_\_ == '\_\_main\_\_':

final\_password = create\_strong\_password()

print("Your unique and secure password is:", final\_password)

Output for the above coding:

Your unique and secure password is.: QNGm1##m6rrv

This Python program uses secrets and string modules to construct strong passwords like password managers, making it exciting. The secrets module generates cryptographically secure passwords, and the string module provides all alphabetic, numerical, and special characters. This basic code loops through 12 passwords with at least one special character and two numbers.

Password generators, for instance, are approachable and useful in the pursuit of data security. (McAfee, 2020). Passwords generated by such systems are more secure because of their randomness and complexity.

(c)

Making use of a safe password generator by rewriting Python code.

Code:

import secrets

import string

#create Strong password

def create\_strong\_password(power\_password=12):

charac = string.ascii\_lowercase + string.ascii\_uppercase

numeric = ''.join(str(i) for i in range(10))

unique\_char = string.punctuation

#Add all the total

entire = charac + numeric + unique\_char

private\_password = ''

personal\_password = False

#Checking for the correct details

while not personal\_password:

private\_password = ''.join(secrets.choice(entire) for \_ in range(power\_password))

if (any(charac in unique\_char for charac in private\_password) and

sum(charac.isdigit() for charac in private\_password) >= 2):

personal\_password = True

return private\_password

def newpassword(pass\_num, power\_password=12):

unique\_secure\_passwords = []

for \_ in range(pass\_num):

distinctive\_password = create\_strong\_password(power\_password)

unique\_secure\_passwords.append(distinctive\_password)

return unique\_secure\_passwords

if \_\_name\_\_ == '\_\_main\_\_':

#Able to have more than 1 password (if required by the user)

pass\_num = 1

Secure\_password = newpassword(pass\_num)

print("These are your unique and secure passwords:")

for distinctive\_password in Secure\_password:

print(distinctive\_password)

Output for the above coding:

These are your unique and secure passwords:

H&t21b1{],`d

Reasons for restructuring the code in bullet points

* Using meaningful names for variables improves the code's readability and comprehension.
* Updated Variable names like “char” to “charac”, “private\_password” to “exclusive\_Password”.
* In this revision, I included a new function called newpassword that, if necessary, may generate many robust passwords all at once. The pass\_num option allows you to select the desired output length of generated passwords. Each of the created passwords is added to a list and then displayed on the screen.
* Code that is well-organized and well-commented is easier to read, both by others and by you in the future.
* Reduced ambiguity and misinterpretation result from better naming and comments.
* Coding Outputs: The created password's purpose must be communicated clearly and concisely in the output message. Change the wording to "These are your unique and secure passwords" for better understanding.

Question 2

Coding:

products = ['laptop', 'mouse', 'webcam', 'keyboard', 'speaker']

updated\_items = []

print(f"We have a list of products here: {products}.")

while True:

item = input("Hello! What do you want to buy? ")

if item not in products:

print("Wrong product! Please try again.")

continue

price\_of\_item = input("How much is it (SGD)? ")

entered\_input = [item, price\_of\_item]

updated\_items.append(entered\_input)

query = input("Do you want to continue? (True/False) ")

if query.lower() != 'True':

break

print(f"This is our updated shopping list: {updated\_items}")

print(f"Final shopping list: {updated\_items}")

Three improvements are suggested to improve code reliability, readability, maintainability, and quality:

1. Name variables based on their function, rather than using generic names like "item" or "entered\_input." Instead of item and price\_of\_item, use desired\_product and product\_price. This simplifies and clarifies the code.
2. Implement input validation to ensure accurate user input. For instance, you can check and correct pricing. This improves dependability by preventing unexpected input from causing issues.
3. Divide concerns into functions: Coding into smaller functions that do certain tasks. One technique accepts user input, validates it, and updates the shopping list. Maintainability improves with modular, comprehensible, testable, and customizable code.

These modifications make the code more reliable, readable, understandable, and high-quality due to clarity and robustness.

References:

Wong Yang (2020, Jun). Coronavirus: NUS students get zero marks for cheating in take-home exam. <https://www.straitstimes.com/singapore/education/coronavirus-nus-students-get-zero-marks-for-cheating-on-take-home-exam>

McAfee (2020, July). Create Strong Passwords with a Password Generator. <https://www.mcafee.com/blogs/privacy-identity-protection/create-strong-passwords-with-a-password-generator/>

Simran Kaur Arora (2023, September). 30 Cool, Easy & Fun Python Projects with Source Code [2023]. https://hackr.io/blog/python-projects