Student Assignment Brief

Contents:

The work you submit for this assignment must be your own independent work, or in the case of a group assignment your own groups' work. More information is available in the 'Assignment Task' section of this assignment brief.

Assignment Information

Module Name: Advanced Algorithms

Module Code: 5002CMD

Assignment Title: Portfolio

Assignment Due: 15th November 2025

Assignment Credit: 10 credits

Word Count (or equivalent): -

Assignment Type:

Percentage Grade (Applied Core Assessment). You will be provided with an overall grade between 0% and 100%. To pass the assignment you must achieve a grade of 40% or above.

Assignment Task

- This is an **individual assignment**. There are **THREE (3)** programs that must be completed and a report to be submitted before the stated due date. The programs will cover a range of data structures and algorithms taught in this module as well as a basic concurrent application.

- You are required to submit a report as well as upload your complete source code (in Python) for all THREE (3) questions to **GitHub**.

- Prepare a documentation (in pdf format) for the programming tasks and submit before the due date. The requirements of the report are explained under each question below.

- There will be a **VIVA session** for each student to allow you to demonstrate your understanding and highlight the originality of your work. You are required to explain how your code works. The VIVA session will help you to show your fulfilment of the learning outcomes.

- Read the important notes and marking rubric carefully to avoid any penalties.

Question 1: Hashing

1. Implement a **Hash Table** that uses **Separate Chaining** as its collision-resolution technique.

2. Develop a local storage system for a **baby products retail shop** based on the data structure created in #1. You must create at least **ONE (1) entity class** that can store the products of the baby shop. Your system must be able to store at least **ONE (1) type of product** from the baby shop. Then, insert a few pre-defined records into the hash table. You can decide the size of the hash table and the number of records to be inserted.

3. Create a **command-line Inventory System** for the local storage system done in #2. Your Inventory System must have an **insert function and a search function**, while the edit and delete function are optional.

4. Compare the performance of searching for records in the hash table **with a one-dimensional array**. For this comparison to work, you must insert the same data into the array and **measure the execution time for the searching**. Analyse the results and provide the explanations of why one is more superior in performance compared with the other.

5. Provide the following discussions in your report:

   i.) Screenshot the code segment which defines the structure of the buckets of the Hash Table. Explain your choice (e.g. simplicity, efficiency, etc) of the structure for the bucket (e.g. array / linked-list).

   ii.) Screenshot the code segment which defines the entity class of a product. Explain the choice of the class attributes and its data types.

iii.) Provide a complete documentation on the experiment done in #4 above. Your documentation must include sample code screenshots, output screenshots, your analysis, and a conclusion.

Question 2: Graph

1. Construct an **unweighted directed graph** data structure. Include the following methods in your graph data structure:
   - **addVertex:** add a new vertex to the graph
   - **addEdge:** connect one vertex with another vertex
   - ***listOutgoingAdjacentVertex***: for a given vertex, this method should list all vertices in which the edges are outgoing from this vertex.

Programming Best Practice (Abstraction):
It is recommended to keep your graph data structure free from any code related to a specific domain/entity (i.e. Person entity). Keep the data type generic for the vertices in your Graph data structure to make your Graph data structure reusable in other contexts. Keeping your code reusable is a good programming practice.

2. Create a Person domain / entity class that represents a single user of a social media app. You may select any relevant attributes to your Person domain / entity class. Examples of attributes are *name*, *gender*, *biography*, *privacy* (*public* / *private* profile), etc.

3. Create the profile for a **minimum of FIVE (5)** people by creating Person objects. Limit the number of profiles to a **maximum of TEN (10)** people.

4. Create a **graph object** that holds the data for a simplified social media app that connects people in a similar way as Instagram. The graph must be able to hold the data of the following and followers for every user. You may design the connections in any way, in terms of who is following who. Mimic the social media app Instagram such that a person can follow another person and can be followed back in return but not necessary.

5. **Design a menu-driven program that allows the user to perform the following actions:**
   - ❖ **Mandatory Features (Compulsory):**
     a) Display a list of all the users' names
     b) View the profile of any one person in detail (Ignore the privacy of the person and display all information in the profile).
     c) View the list of followed accounts of a particular person (This involves listing all the outgoing edges of a particular vertex).
     d) View the list of followers of a particular person (You may implement this in any way workable and efficient).

❖ **Advanced Features (Optional. You may select any one or combinations to implement):**

a) Add a user profile on-demand (This involves creating a new vertex on-demand in the graph).

b) For people with the **private settings**, their profile only shows the name, and the other details are hidden. For people with the **public settings**, their profile shows the name and all other details (This involves showing the details of a single vertex object).

c) Allow a user X to **follow** another user Y on-demand (This involves creating a new outgoing edge from X to Y on-demand).

d) Allow a user X to **unfollow** another user Y on-demand (This involves removing an outgoing edge from X to Y on-demand).

If you wish to develop a GUI (Graphical User Interface) instead of a command-line interface (CLI), you may do so, but it will **not result in additional marks** due to it not being under the learning outcomes.

6. Provide the following discussions in your report.

   i.) Screenshot the code segments that handle the mandatory features and provide brief explanations on how the code works.

   ii.) [For those who had done any of the extra features] Screenshot the code segments that handle the extra features that you implemented and provide brief explanations on how the code works.

   iii.) Provide **screenshots for the output** for all the features implemented. Use test data to demonstrate that your program is working properly. Include brief explanations whenever necessary if they can clarify your points.

   Some sample outputs for Question 2 are provided at the end of this document. You may use it as a guideline but not required to follow exactly the sample output. Use your creativity to produce a better output than the one shown to you.

Question 3: Concurrent process

1. Python program supports multithreading. Discuss whether in Python, it is concurrent processing or parallel processing.

2. Write a function that performs the calculation of a Factorial based on a given number. Derive the Big-O of this function by calculating the number of primitive operations and explain its time complexity.

3. Using multithreading:

- Write a program to calculates the factorials: 50!, 100! and 200!. Use multithreading and create a separate thread to handle each set of the operation (i.e. you must create 3 separate threads to handle the 3 sets, 1 thread for each set).

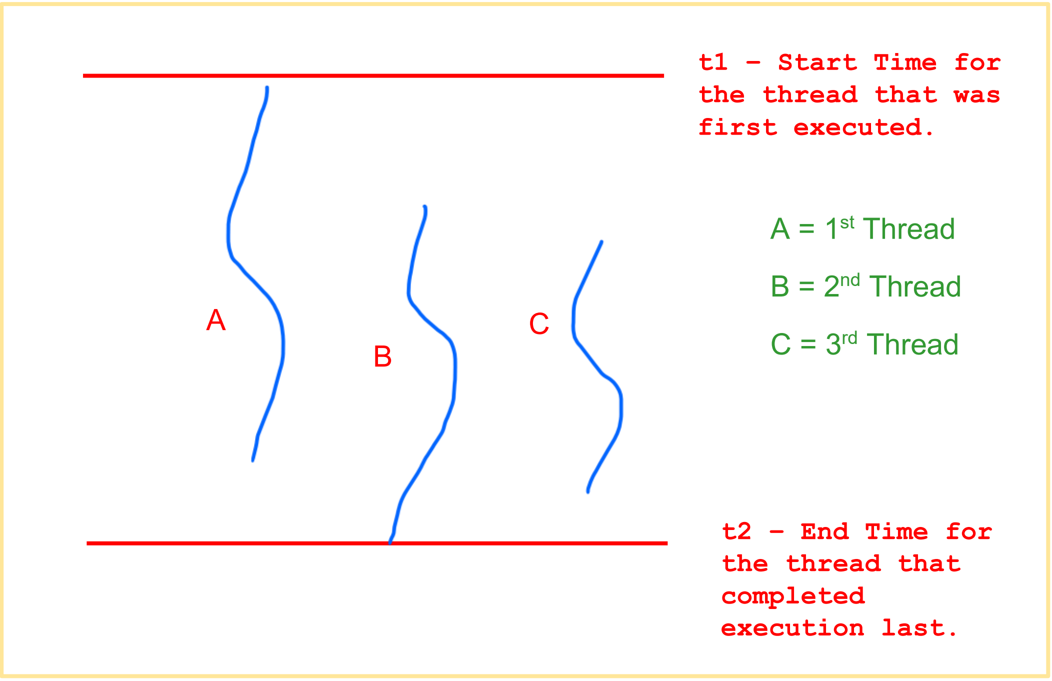
- Measure the time taken (in nanosecond) to complete the operation of generating the three factorial numbers using multithreading. The total time can be calculated with the following formula:

  Time_Elapsed = End_Time_Of_Thread_Finished_Last – Start_Time_Of_Thread_That_Started_First

  Learn how to get the current system time in nanosecond for Python from the link below:

https://docs.python.org/3/library/time.html

  The illustration below shows how to calculate the total time taken for the 3 sets of operation in separate threads.


t1 – Start Time for the thread that was first executed.
A = 1st Thread
B = 2nd Thread
C = 3rd Thread
A
B
C
t2 – End Time for the thread that completed execution last.

  T = t2 – t1

- Perform testing in 10 rounds for generating the same numbers and then observe the time taken (T) for each round as well as the average time taken (Average for T) for all the 10 rounds. Your program should show the T value in nanosecond for each round of operation. Format the output as desired but make the output clear and easy to read.

4. Perform the same operations as #3 (i.e. generate the factorial for 50!, 100!, and 200!), this time **without multithreading**. Calculate the total time taken for each round and the average time taken and display the results in your program output.

5. Provide the following discussions in your report:

   i.) Provide a screenshot of the code for the function that calculates the factorial number. Derive the Big-O of the function and explain its time complexity.

   ii.) Based on the experiments that you had done in Step #3 and Step #4 above, make a comparison and provide an analysis of your results and findings. Explain whether multithreading shortens the time taken for the tasks. You must include screenshots of the output with some sample data. Your screenshots must show the total and average time taken for each experiment.

   iii.) Screenshot the code segment that handles multithreading and explain how the code works.

   iv.) Discuss why multithreading did or did not shorten the time taken for this experiment and suggest an example of situation where multithreading will improve the performance.

Challenges Faced and Personal Reflections in Doing This Assignment

In **no more** than 2 pages, write your personal experience while doing this assignment. You may talk about the challenges faced, overcoming the obstacles, solving problems, picking up new skills, managing time, developing new habits, cultivating discipline, etc.

You may share anything that is sharable and express your feelings of going through the process of completing this assignment.

Use of Generative AI in this assignment:

- **Amber (AI Used to assist)** – Generative AI tools such as *ChatGPT*, *Gemini*, or *Copilot* **may be used in the learning process** to understand the concepts but **should not be used to generate any part of the writings in your report**. You may use the codes generated by tools like ChatGPT, but you **must understand how the code works**, which you will be tested during the VIVA. As for the report writing, no parts of the report should be written by AI. The main intention of preventing the use of Generative AI in such way is to **avoid over-reliance** on these AI tools which hinder the learning process and preventing you from achieving the learning outcomes and exercising your cognitive and psychomotor skills required to pass the coursework. Your knowledge and skills acquired in the process of doing this assignment will be tested during VIVA in which you are required to demonstrate your ability to answer the questions and show how far you are at in achieving the learning outcomes. Your report contents will also be tested with tools that can detect AI-generated answers.

Submission Instructions:

Method of submission:

1. Submit your report via Canvas before the deadline.

2. You are required to upload only **ONE (1) PDF** document onto **Canvas**. You DO NOT NEED to submit your source code on Canvas. However, you must **upload your complete source code to GitHub** and provide this link on the first page of the cover of your report. FAILING TO provide the link to GitHub, or if your code **cannot be accessed** due to failure on your part might result in you getting zero for this assignment and **FAILING THIS COURSEWORK**.

3. A VIVA session of up to a maximum of 15 minutes per person will be scheduled at the end of the semester to test how far you have come to achieve the learning outcomes. Failing to register or turn up for the VIVA session might result in you FAILING THIS COURSEWORK as well.

4. In your report, label everything clearly and provide suitable headers for every section and question. The General format for the report is as follow:
   - Cover Page (With GitHub Link to your Source Code)
   - TurnItIn Result Summary Page
   - Table of Contents
   - Question 1 Contents
   - Question 2 Contents
   - Question 3 Contents
   - Challenges faced and personal reflections
   - Appendix (Optional)
   - References

Marking and Feedback

How will my assignment be marked?

Your assignment will be marked by the module leader.

How will I receive my grades and feedback?

Provisional marks will be released once internally moderated.

Feedback will be provided by the module leader alongside grades release.

Marks and feedback can be accessed by going to the Grades on Canvas system.

Your provisional marks and feedback should be available by end of the semester.

What will I be marked against?

Details of the marking criteria for this task can be found at the bottom of this assignment brief.

Assessed Module Learning Outcomes

The Learning Outcomes for this module align to the marking criteria which can be found at the end of this brief. Ensure you understand the marking criteria to ensure successful achievement of the assessment task. The following module learning outcomes are assessed in this task:

1. Evaluate the complexity and efficiency of algorithms for solving a range of problems
2. Apply algorithms and data structures to solve novel problems
3. Demonstrate solutions to intractable problems by considering computational complexity and algorithm theory.
4. Analyse the issue of data consistency and multi-threading in a concurrent application.

Assignment Support and Academic Integrity

Spelling, Punctuation, and Grammar:

You are expected to use effective, accurate, and appropriate language within this assessment task.

Academic Integrity:

The work you submit must be your own, or in the case of groupwork, that of your group. All sources of information need to be acknowledged and attributed; therefore, you must provide references for all sources of information and acknowledge any tools used in the production of your work. We use detection software and make routine checks for evidence of academic misconduct.

It is your responsibility to keep a record of how your thinking has developed as you progress through to submission. Appropriate evidence could include version-controlled documents, developmental sketchbooks, or journals. This evidence can be called upon if we suspect academic misconduct.

If using Artificial Intelligence (AI) tools in the development of your assignment, you must reference which tools you have used and for what purposes you have used them. This information must be acknowledged in your final submission.

Unable to Submit on Time?

Student will be graded as ZERO if they are unable to submit on time.

Administration of Assessment

Module Leader Name: Karen Khor Jia Yun

Module Leader Email: karen.khor@newinti.edu.my

Assignment Category: Portfolio

Attempt Type: [Standard / Resit / Deferral]

Component Code: CW

Assessment Marking Criteria

Criteria	Weightage
Q1 Code	10%
Q2 Code	10%
Q3 Code	10%
Q1 Report Writing	10%
Q2 Report Writing	10%
Q3 Report Writing	10%
VIVA	40%
TOTAL	**100%**

Detailed Assessment Criteria

	Fail **0 to 29%**	**Fail** **30 to 35%**	**40 to 49%** **[Basic]**	**50 to 59%** **[Average]**	**60 to 69%** **[Good]**	**70 to 79%** **[Excellent]**	**80 to 100%** **[Outstanding]**
	0 – 15% Failed by submitting a subpar work that barely has the structure of the basic things needed to pass. **16 – 29%** Failed by submitting a subpar work. Work submitted shows that the student has no idea how to fulfil the given tasks.	**30 – 32%** The work failed to fulfil the basic requirements but can be passed if more work is put on the work. **33 – 35%** The work failed to fulfil the basic requirements but can be passed if only a little more work is put on it.	**40 – 45%** Passed by fulfilling the basic requirements in a weak way **46 – 49%** Pass by fulfilling the basic requirements in a strong way	**50 – 55%** Basic requirements are fulfilled and on top of that, is better. **56 – 59%** Basic requirements are fulfilled and work done is average and almost reaching “Good” level.	**60 – 65%** Work done is above average and is worth being labelled as “Good”. **66 – 69%** Work done has far exceeded average and almost reaching excellent but missed slightly.	**70 – 75%** Work done is excellent, and worth being commended and praised. **76 – 79%** Work done close to being outstanding but missed slightly.	**80 – 89%** Achieved with some room for improvements **90 – 100%** Achieved with perfection and almost flawless

Q1 Code (Completeness and Error-Free) Weighting: 10%	This part is missing in the report. OR The GitHub link is inaccessible, or the link is not provided in the cover page of the report. OR Code not found or missing part of the code in the GitHub link provided.	The GitHub link was provided on the cover page of the report and is accessible. The code failed to run, or they compiled with errors. OR Failed to fulfil all the basic requirements in the question.	The code performs all the basic tasks required in the question. **No output** or the output produced is very minimal and might lack crucial information. No abstraction applied in code. No chance of code reuse in the design of data structures and algorithms. A lot of hardcoding found in the work submitted.	The code performs all the basic **tasks** required in the question. The **output** shows all the required information with very basic or no formatting. No abstraction applied in code. No chance of code reuse in the design of data structures and algorithms. Some level of hardcoding found in the work submitted.	The code performs more than the basic **tasks** required in the question. The **output** shows all the required information with good formatting Some level of abstraction applied in the code. Certain parts of the code can be reused in other programs. No unnecessary hardcoding.	The code performs more than the basic tasks required in the question. The **output** shows all the required information with good formatting A good level of abstraction applied in the code. Most of the data structure code can be reused in other programs. No unnecessary hardcoding.	Clean code, well-commented, well-indented and good readability. The code performs more than the basic **tasks** required in the question. The output shows all the required information with excellent formatting and demonstrate extensive effort put into the work. Excellent level of abstraction applied in the code. All the data structures can be reused in other programs. No unnecessary hardcoding.
Q2 Code (Completeness and Error-Free) Weighting: 10%							
Q3 Code (Completeness and Error-Free) Weighting: 10%							

Q1 Writing Weighting: 10% **Q2 Writing** Weighting: 10% **Q3 Writing** Weighting: 10%	This part is missing, or the work submitted is inadequate and incomplete.	Missing code explanation or output screenshots. Discussions are in a very rudimentary form and is below the acceptable level for passing.	**Code Explanation:** Explanation of the code is merely narrating the code in words but not explaining. **Output and Discussions:** Discussions or output shown lack clarity and/or details and missing crucial points.	**Code Explanation:** Able the explain the code with average breadth and depth **Output and Discussions:** Able to show the program output clearly and provide a basic level of discussion on the topic with adequate coverage	**Code Explanation:** Able the explain the code with average breadth and depth **Output and Discussions:** Able to show the program output clearly and provide an average level of discussion on the topic with good coverage	**Code Explanation**: Able the explain the code with **proper breath** but lacking in depth. Coverage is adequate **Output and Discussions**: Able to show the program output clearly and provide a good level of discussion with excellent coverage	**Code Explanation:** Able the explain the code with proper depth and breath. Not over-explaining but with adequate coverage with enough depth. **Output and Discussions:** Able to show the program output clearly and provide a high-quality discussion on the topic.
VIVA Ability to demonstrate mastery over own work Weighting: 20%	Failed to turn up for the VIVA or turned up late that caused insufficient time for the student to demonstrate this ability.	**Failed** to explain the code submitted even for the basic things.	Only able to explain the code submitted for very basic things. Unable to explain more complex part of the code.	Able to explain some of the code submitted but showed uncertainties in some of the code submitted.	Able to explain most of the code submitted with **moderate clarity** and confidence. Might have made some mistakes in accuracy or correctness in some parts of the explanation.	Able to explain most of the code submitted with **high clarity** and confidence. Might have made some mistakes in accuracy or correctness in some parts of the explanation.	Able to explain all the code submitted with high clarity, confidence, and accuracy.

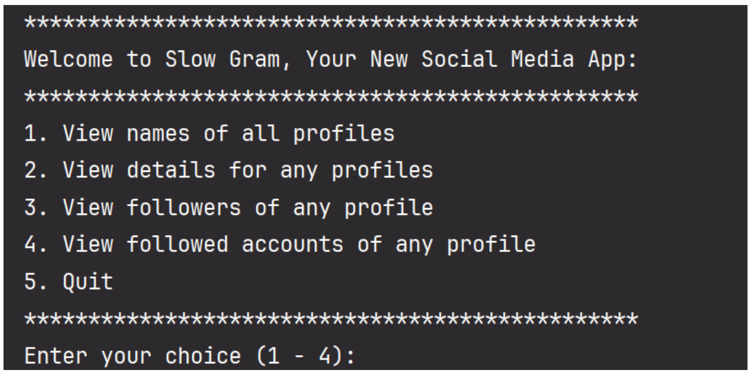
VIVA Ability to provide a solution for different situations Weighting: 10%	Failed to turn up for the VIVA or turned up late that caused insufficient time for the student to demonstrate this ability. OR Failed to provide a solution for a new problem.	Failed to provide a solution to the new problem, but able to give a rough idea near to a possible solution.	Able to provide an average solution to the new problem, but unable to express the solution with further details.	Able to provide an average solution to the new problem and able to express the solution with adequate details.	Able to provide a good solution to a new problem but unable to express the idea with adequate details.	Moderately articulate in expressing the solution and demonstrated good problem-solving skills.	Very articulate in expressing the solution and demonstrated an excellent out-of-the-box thinking that can create novel solutions to new problems.
VIVA Ability to demonstrate evidence of basic skills required to produce the work submitted Weighting: 10%	Failed to turn up for the VIVA or turned up late that caused insufficient time for the student to demonstrate this ability.	Failed to demonstrate even **basic** skills required to produce the work submitted. Showed very **weak understanding** in the work submitted.	Showed **a lot** of signs of uncertainties in carrying out the basic tasks required to produce the work submitted.	Showed **some** signs of uncertainties in carrying out the basic tasks required to produce the work submitted.	Showed confidence and some proficiency in carrying out the basic tasks required to produce the work submitted.	Showed confidence and high proficiency in carrying out the tasks required to produce the work submitted.	Demonstrated outstanding skills in programming, which is higher than expected from a college level student.

Note:

Submitting a long report with many pages or many paragraphs **does not guarantee high marks**. The conciseness, relevance, accuracy, depth, coverage and quality are several of the qualities assessed on your report. Some reports might have many paragraphs but is low in conciseness. Some may have many paragraphs, but the contents are largely irrelevant. So, when you are doing the report for this assignment, focus on the quality and not so much on the number of words or number of pages. Furthermore, during VIVA you will be asked on selected parts of your report.

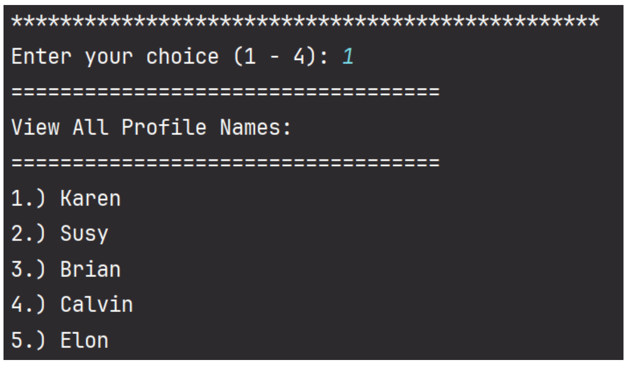
Question 2 Sample Program Output

Program Main Menu:



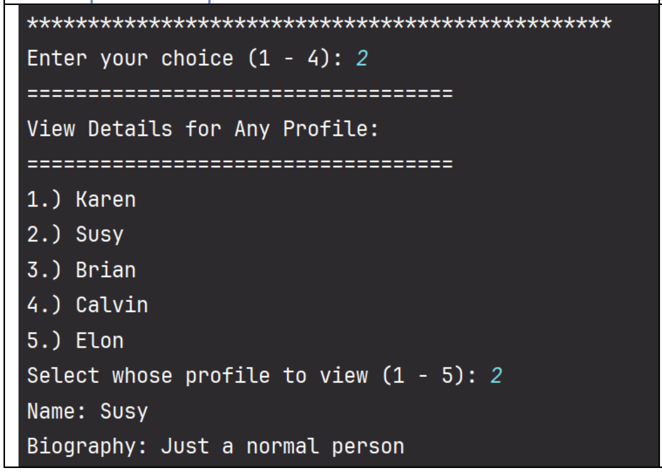
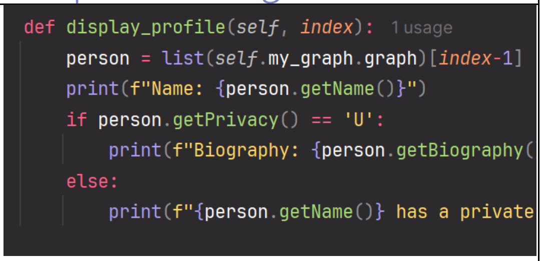
```
**************************************************
Welcome to Slow Gram, Your New Social Media App:
**************************************************
1. View names of all profiles
2. View details for any profiles
3. View followers of any profile
4. View followed accounts of any profile
5. Quit
**************************************************
Enter your choice (1 - 4):
```

View all profile names



```
************************************************
Enter your choice (1 - 4): 1
===================================
View All Profile Names:
===================================
1.) Karen
2.) Susy
3.) Brian
4.) Calvin
5.) Elon
```

View Details for Any Profile

Sample Output:	Sample Code Segment:
	

Sample Output:

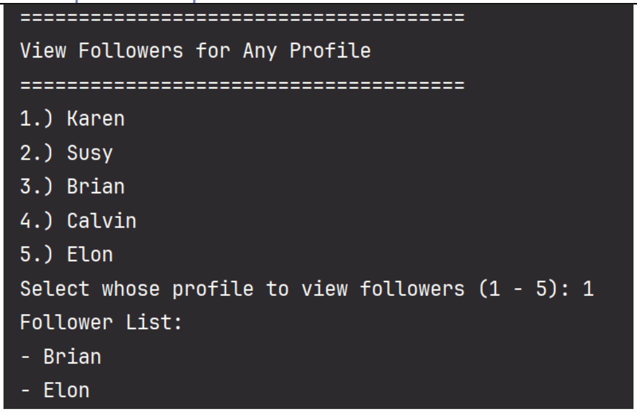
```
************************************************
Enter your choice (1 - 4): 2
===================================
View Details for Any Profile:
===================================
1.) Karen
2.) Susy
3.) Brian
4.) Calvin
5.) Elon
Select whose profile to view (1 - 5): 2
Name: Susy
Biography: Just a normal person
```

Sample Code Segment:

```python
def display_profile(self, index):  1 usage
    person = list(self.my_graph.graph)[index-1]
    print(f"Name: {person.getName()}")
    if person.getPrivacy() == 'U':
        print(f"Biography: {person.getBiography(
    else:
        print(f"{person.getName()} has a private
```

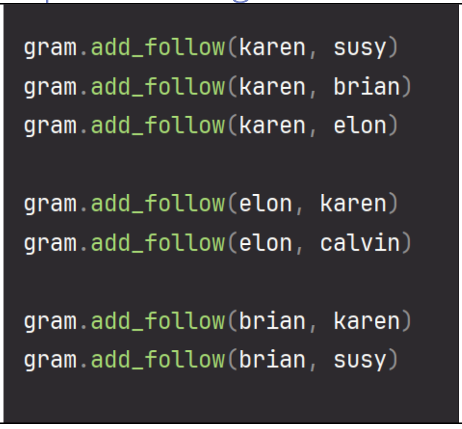
View Followers for Any Profile

Sample Output



```
======================================
View Followers for Any Profile
======================================
1.) Karen
2.) Susy
3.) Brian
4.) Calvin
5.) Elon
Select whose profile to view followers (1 - 5): 1
Follower List:
- Brian
- Elon
```

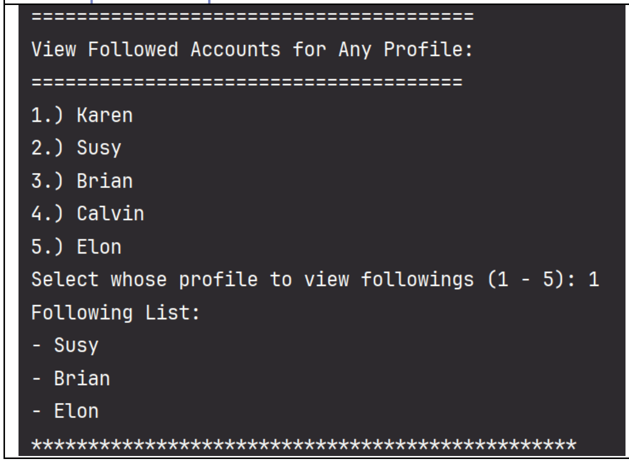
Sample Code Segment



```
gram.add_follow(karen, susy)
gram.add_follow(karen, brian)
gram.add_follow(karen, elon)

gram.add_follow(elon, karen)
gram.add_follow(elon, calvin)

gram.add_follow(brian, karen)
gram.add_follow(brian, susy)
```

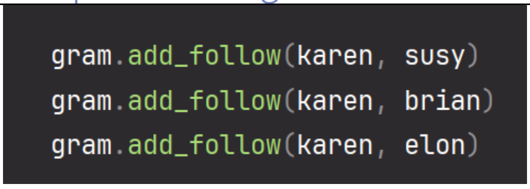
View Followed Accounts for Any Profile

Sample Output:



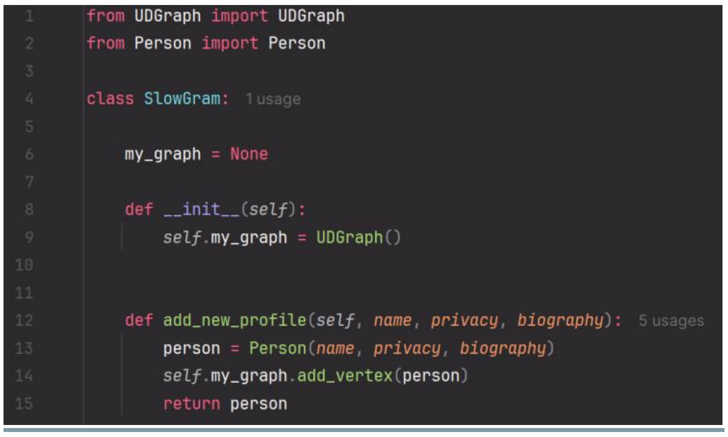
```
======================================
View Followed Accounts for Any Profile:
======================================
1.) Karen
2.) Susy
3.) Brian
4.) Calvin
5.) Elon
Select whose profile to view followings (1 - 5): 1
Following List:
- Susy
- Brian
- Elon
***********************************************
```

Sample Code Segment:



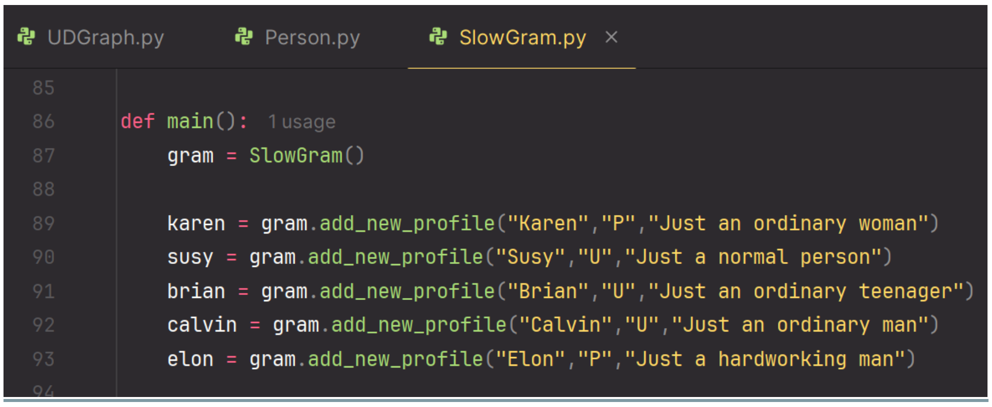
```
gram.add_follow(karen, susy)
gram.add_follow(karen, brian)
gram.add_follow(karen, elon)
```

Sample Code for the Simplified Social Media App



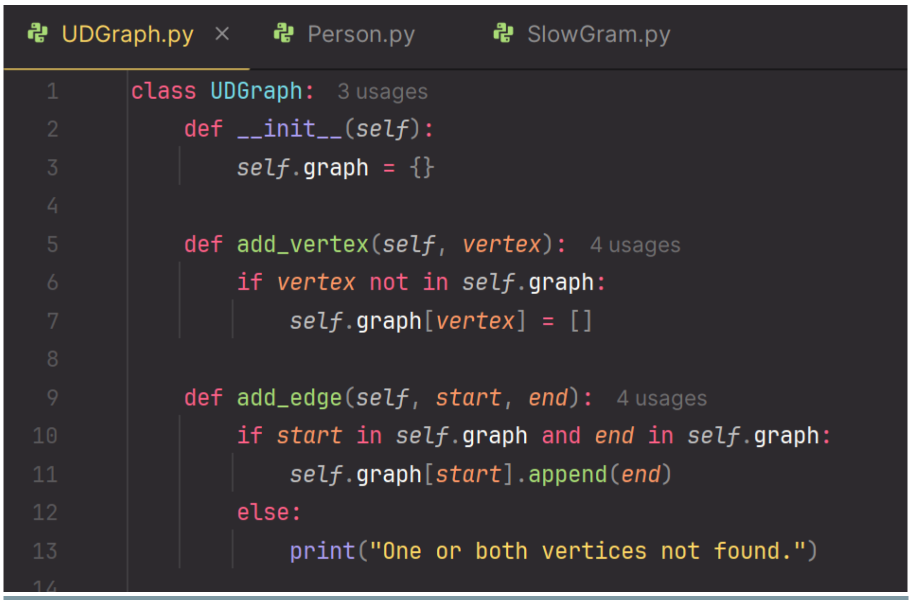
```
from UDGraph import UDGraph
from Person import Person

class SlowGram:  1 usage

    my_graph = None

    def __init__(self):
        self.my_graph = UDGraph()


    def add_new_profile(self, name, privacy, biography):  5 usages
        person = Person(name, privacy, biography)
        self.my_graph.add_vertex(person)
        return person
```

Sample Code for the Driver Program



```
def main():  1 usage
    gram = SlowGram()

    karen = gram.add_new_profile("Karen","P","Just an ordinary woman")
    susy = gram.add_new_profile("Susy","U","Just a normal person")
    brian = gram.add_new_profile("Brian","U","Just an ordinary teenager")
    calvin = gram.add_new_profile("Calvin","U","Just an ordinary man")
    elon = gram.add_new_profile("Elon","P","Just a hardworking man")
```

Sample Code for the Graph Data Structure



```
class UDGraph:  3 usages
    def __init__(self):
        self.graph = {}

    def add_vertex(self, vertex):  4 usages
        if vertex not in self.graph:
            self.graph[vertex] = []

    def add_edge(self, start, end):  4 usages
        if start in self.graph and end in self.graph:
            self.graph[start].append(end)
        else:
            print("One or both vertices not found.")
```