Computer Science – ATAR Year 12

Task 2 – Unit 3

Assessment type: Project: Programming

Conditions: Time for task 4 weeks. Due at the end of week 3, Term 2.

Task weighting
20% of the school mark for this pair of units

Scenario:

Papa Pizza, a local Italian pizza shop in Subiaco, has approached you to develop a software solution. Your task is to program an application capable of receiving and processing pizza orders. This program should be able to calculate the cost for each individual pizza order and keep track of the cumulative totals, providing a daily summary of the quantities sold for each pizza variety. The menu includes:

Pepperoni	$21.00
Chicken Supreme	$23.50
BBQ Meatlovers	$25.50
Veg Supreme	$22.50
Hawaiian	$19.00
Margherita	$18.50

Additionally, the program must account for specific discounts and surcharges:

- For homedelivery - a delivery fee of $8.00 is added to the final bill.
- Orders exceeding $100, as well as those made by loyalty card members, are eligible for a 5% discount.
- The total bill amount should include a Goods and Services Tax (GST) calculated at 10% of the amount after applying any discounts and surcharges.

What you need to do

To successfully develop your Pizza Ordering Application, you will need to make use of Object-Oriented Programming (OOP) features:

Modular Approach: Employ functions and classes to structure your code efficiently.

Classes and Objects: Design and implement classes with attributes and methods. Instantiate objects from these classes.

Inheritance: Use inheritance to extend classes, enabling code reuse and the establishment of hierarchical relationships.

Data Structures: Implement one- and two-dimensional arrays and dictionaries to manage and organize data effectively.

Control Structures: Utilise a variety of control structures, including sequence, selection, and iteration, to control the flow of your program.

Best Practices: Apply good programming practices to ensure your code is readable, maintainable, and efficient.

Your final project needs to be complex enough to demonstrate your understanding of software development.

Part 1

1.1: Investigate

Project Development Steps and Timeline:

- Break down the steps required to develop the PapaPizza Ordering System and establish a timeline for the completion of each phase.

Problem Outline:

- Outline the purpose and objectives of the PapaPizza Ordering System.

Problem Description:

- Describe the system, detailing:
    - The objective:
    - The method for calculating the cost of orders, including applying discounts and surcharges.
    - The process for tracking orders and daily sales.
    - Any additional features or functionalities that will be included in the system.

Requirements List:

- Compile a detailed list of requirements for the PapaPizza Ordering System, categorised as follows:
    - **Functional Requirements:** Order placement and processing, cost calculation, sales tracking, and discount application.
    - **Non-Functional Requirements:** Usability, performance (speed and reliability), security (data protection and secure payment processing), and legal.

1.2: Design

Using pseudocode, write an algorithm showing the core logic for PapaPizza Ordering System that efficiently processes orders, calculates costs, applies discounts and surcharges, and tracks daily sales.

Note: this algorithm will not resemble your final, fully functioning code. It should simply demonstrate how the core logic for the game works.

Structured algorithm

Your algorithm should include:

- Use of pseudocode for representing algorithms.
- Creation of code with effective structure through:
    - A modular approach (including functions and classes).
    - Parameter passing.
    - Use of stubs to represent incomplete functions.
- **Apply the following using pseudocode:**
    - Data types as variables.
    - Modular coding using functions, parameters, and arguments.
    - Different types of operators: arithmetic, relational, and logical.
    - Arrays:
        - One-dimensional arrays.
        - Two-dimensional arrays.
    - Dictionaries.

Create:

1.2.2:

- Test your algorithms using trace tables and appropriate test data.

Part 2

Develop:

- Using a modular approach and incorporating Object-Oriented Programming (OOP) principles, construct your programme in Python. Ensure you use good programming practices as indicated in the syllabus.

Part 3

Testing:

- Develop a test plan for your program to ensure that it is fully tested and documented. Your test plan should include appropriate test data, type and range checks. You should document this test plan in a table outlining the input, the expected output and what actually happened.

Evaluate

- Reflect on the success of your system and how well it meets the system requirements. To perform your user acceptance testing, you should:
  - consider how well your program meets the requirements you developed in Part 1
  - consider what aspects of your program could be improved and the quality of the user experience. You might want to get your peers to test your program and give feedback.
  - document any known bugs and/or limitations with your program and explain how they impact the performance of the system
    Note: as part of your evaluation, you should consider any changes that you have made to your design and justify these changes.
- Reflect on the process you followed to develop your system and how you could improve this process. Some aspects you should consider include:
  - what worked well?
  - what didn't work well?
  - what would you do differently next time?
- Document the sources you used to get information about how to develop your system, including all websites and textbooks.

Marking Criteria

Description	Marks
Development Schedule	
Breaks down the project planning into a series of meaningful steps and a realistic timeline for completing each step has been included	2
Breaks down the project into a limited series of steps with some attempt at showing a timeline.	1
Subtotal	**/2**
Problem Outline	
Accurately outlines the system's purpose and objectives	2
Basic outline of the system's purpose and objectives	1
Subtotal	**/2**
Problem Description	
Detailed explanation of the PapaPizza ordering system, including objectives, order process, cost calculation, tracking, and additional features:	5
Clearly explains the Pizza Ordering System, its functionality, and user interaction, with reference to advanced features.	4
Describes the Pizza Ordering System, its functionality, and how it is used, with reference to some advanced features.	3
Gives a limited description of the system with some reference to its use.	2
Gives a limited description of the system that is unclear and/or incomplete.	1
Subtotal	**/5**
Requirements	
Provides a clear and detailed list of requirements that fully meet the needs of the problem description. Suitably classifies requirements.	4
Completes a list of requirements that meet the needs of the problem description. Classifies requirements.	3
Provides a list of requirements that mostly meet the needs of the problem description. Partially classifies requirements.	2
Provides an incomplete list of requirements that meet some of the needs of the problem description. Makes a limited attempt at classifying requirements.	1

Description	Marks
Subtotal	**/4**
Algorithms	
Provides a completed algorithm in pseudocode that accurately represents the core logic of the Pizza Ordering System, using correct modules, symbols, and/or syntax.	5
Provides a completed algorithm in pseudocode that represents the core logic of the Pizza Ordering System, using correct modules, symbols, and/or syntax.	4
Provides a mostly complete algorithm in pseudocode that provides a partial representation of the core logic of the Pizza Ordering System, with symbols and/or syntax that is mostly correct.	3
Provides an algorithm with a partial solution to processing orders, with some errors in syntax, logic, and/or symbols being used.	2
Provides a partially correct algorithm and/or uses incorrect symbols/syntax.	1
Subtotal	**/5**
Develops an accurate algorithm that contains no logic errors, and demonstrates the use of a range of control structures	3
Develops an accurate algorithm that may contain minor logic errors, and uses a range of control structures.	2
Develops an algorithm that contains logic errors, and uses a minimal range of control structures.	1
Subtotal	**/3**
Trace Tables	
Completes comprehensive algorithm logic testing using appropriate test data in trace table provided	3
Partially tests algorithm using trace table using appropriate test data	2
Partially tests algorithm with trace table, using incorrect format and/or incomplete test data.	1
Subtotal	**/3**
Description	**Marks**

Description	Marks
Use of OOP and Programming Structures	
Makes consistent and appropriate OOP concepts, including classes, objects, inheritance, and polymorphism. Consistent use of control structures and data types:	5
Makes appropriate use of OOP principles with appropriate control structures and data types:	4
Makes use of OOP concepts, although may not use most appropriate structures at times. Attempts to make appropriate use of a variety of data types for variables, including some use of arrays and or dictionaries.	3
Attempts to use OOP concepts Makes limited use of data types with variables and some attempt at using arrays appropriately.	2
Makes minimal use of OOP concepts, with inappropriate use of different data types. Provides arrays and or dictionaries that are not used, or are used inappropriately, and do not serve required purpose.	1
Subtotal	**/5**
Good programming practice	
Appropriately structures code, making effective use of modularisation, Object-Oriented Programming (OOP) principles, parameter passing, appropriate naming conventions, use of white space, and inclusion of stubs for incomplete functions	5
Mostly structures code appropriately, utilising modularisation and OOP principles, parameter passing, with mostly appropriate naming conventions, adequate use of white space, and some stubs for future implementation	4
Creates simplistic code with modularisation and some use of OOP principles, parameter passing. Generally uses appropriate naming conventions, moderate use of white space, and minimal stubs for incomplete functions	3
Attempts to structure code with basic modularisation and limited use of OOP principles, limited parameter passing. Some naming conventions are used, though they may be inconsistent, with minimal use of white space and few or no stubs for incomplete functions	2
Produces poorly structured code with minimal or no use of modularisation or OOP principles, inconsistent and/or not meaningful naming conventions, inadequate use of white space, and lacks stubs for incomplete functions	1
Subtotal	**/5**
Uses accurate and useful comments throughout the code to explain the purpose of modules where necessary.	3

Description	Marks
Uses comments that help make code readable.	2
Makes limited use of comments throughout code.	1
Subtotal	**/3**
Functionality	
Develops an effective and efficient OOP-based program for the PapaPizza project with minimal bugs	4
Develops an effective OOP-based program for the PapaPizza project, but may contain some bugs	3
Develops an OOP-based program for the PapaPizza project with a significant number of bugs	2
Partially completes program implementing minimal system requirements.	1
Subtotal	**/4**
Test Plan	
Completes a detailed test plan and documents testing of the PapaPizza Ordering System. Considers all possible inputs and ordering scenarios	5
Completes a test plan and documents testing of the PapaPizza Ordering System. Considers a wide range of inputs and ordering scenarios.	4
Completes a test plan and documents testing of the PapaPizza Ordering System. Considers some possible inputs and ordering scenarios.	3
Completes and documents a partial test plan for the PapaPizza Ordering System. Considers limited inputs and ordering scenarios	2
Develops a minimal test plan and/or provides minimal documentation of testing for the PapaPizza Ordering System	1
Subtotal	**/5**
User Acceptance Testing	
Demonstrates a detailed evaluation of how the PapaPizza Ordering System meets the requirements identified in Part 1, including an in-depth discussion of the user experience	5
Evaluates how the PapaPizza Ordering System meets the system requirements, with a comprehensive discussion of the user experience	4

Description	Marks
Completes a partial evaluation of how the PapaPizza Ordering System meets the requirements, with a basic discussion of the user experience	3
Provides a limited evaluation of how the PapaPizza Ordering System meets the system requirements	2
Completes a superficial evaluation of the program and how it meets the system requirements.	1
Subtotal	**/5**
Provides a detailed discussion of potential improvements for the final PapaPizza Ordering System product and thoroughly documents any bugs and/or limitations	5
Describes identified bugs and/or limitations with suggestions for how the final PapaPizza Ordering System product could be improved	4
Identifies bugs and/or limitations of the PapaPizza Ordering System, without detailed analysis of their impact on the final product	3
Makes an attempt to identify bugs and/or limitations of the PapaPizza Ordering System, with minimal discussion on improvements	2
Subtotal	**/4**
Retrospective	
Completes a detailed evaluation of the development process for the PapaPizza Ordering System and suggests potential future impacts	3
Completes an evaluation of the development process used for the PapaPizza Ordering System, including some suggestions for future impacts	2
Completes a minimal evaluation of the development process for the PapaPizza Ordering System with superficial comments on the process and potential future impacts	1
Subtotal	**3**
Total	**/61**