



## DESCRIPTION OF COURSEWORK

Course Code	CST209
Course Name	Object-Oriented Programming-C++
Lecturer	Venantius Kumar Sevamalai / Geetha Kanaparan
Academic Session	2024/04
Assessment Title	Final Project

### A. Introduction/ Situation/ Background Information

This objective of this final project is to create a functional C++ program by applying object-oriented programming knowledge and other relevant algorithms to solve the problem described in Section F.

### B. Course Learning Outcomes (CLO) covered

At the end of this assessment, students are able to:

CLO 3 Apply knowledge to write algorithms with Object-Oriented and C++ by using class, object, inherit, and polymorphic.

### C. University Policy on Academic Misconduct

1. Academic misconduct is a serious offense in Xiamen University Malaysia. It can be defined as any of the following:

- Plagiarism** is submitting or presenting someone else's work, words, ideas, data or information as your own intentionally or unintentionally. This includes incorporating published and unpublished material, whether in manuscript, printed or electronic form into your work without acknowledging the source (the person and the work).
- Collusion** is two or more people collaborating on a piece of work (in part or whole) which is intended to be wholly individual and passed it off as own individual work.
- Cheating** is an act of dishonesty or fraud in order to gain an unfair advantage in an assessment. This includes using or attempting to use, or assisting another to use materials

that are prohibited or inappropriate, commissioning work from a third party, falsifying data, or breaching any examination rules.

2. All assessments submitted must be the student's **own** work, without any materials generated by AI tools, including direct copying and pasting of text or paraphrasing. Any form of academic misconduct, including using prohibited materials or inappropriate assistance, is a serious offense and will result in a zero mark for the entire assessment or part of it. If there is more than one guilty party, such as in case of collusion, all parties involved will receive the same penalty.

#### D. Instruction to Students

1. This is an individual project.
2. ~~Submit a softcopy~~ (**zipped file containing program and your report**) of the project via Moodle. The report should be submitted in **PDF** format. *Note that the C/C++ program should run on **CodeBlocks**. Include citations as comments in your code if you have used codes from any online or published resource.*
3. Submission deadline: **15 July 2024 (Monday), before 11.59 pm**
4. **Late submissions will receive a 0 mark.**
5. Your report should contain the following information:
  - a. Cover page with your Name and your Student ID.
  - b. Answers for the assignment task in Section F.
  - c. List of references (this includes codes taken from online or published resources - these must be cited and referenced accordingly).

#### E. Evaluation Breakdown

No.	Component Title	Percentage (%)
<b>Part 1: Food Ordering Application</b>		
a.	Classes and objects	15
b.	Inheritance	10
c.	Polymorphism	10
d.	STL	10
e.	Files	5

f.	Exception handling	5
g.	Quality of implementation	15
<b>Part 2: Project Documentation</b>		
a.	UML Diagrams	10
b.	STL Explanation	10
c.	Inheritance and polymorphism explanation	10
	<b>TOTAL</b>	<b>100</b>

## F. Task(s)

### Part 1: Building a Food Ordering Application (70 Marks)

In this project, you must apply knowledge of C++ object-oriented programming to create a program for a food ordering application.

The program should meet the following requirements:

- Login** – when the program starts, the system should allow a user to login. There should be a minimum of 5 users registered in the system. Include all the necessary variables/data fields that is required for using the food ordering application.
- Main selection menu** – once the user has logged in, a selection menu should appear showing several options for the users to interact with the program. Do ensure that the selection menu continues to loop until the user inputs an option to terminate the loop and exit from the program.
- New food order** – the user can make a food order from a list of food items. You may assume that the user is able to order up to 5 food items in a single order from a single restaurant. The list of food items should contain a minimum of 5 food categories/cuisines (eg. Western, Arabic, Chinese, etc.). There should be at least 1 restaurant for each food category/cuisines. In each restaurant, there should be a minimum of 5 food items to select from. (eg. if food category/cuisine western is selected and Restaurant ABC is selected, then the food items from this restaurant may be chicken chop, fish and chips, garlic bread etc.). The customer should be allowed to enter the quantity of each food item to order and add any special instructions to the restaurant. In addition, for some food items, customers may be allowed to select their preference for the food item (eg. cod fish or haddock fish for type of fish and chips etc.)

- d) **Delivery** – once the customer has selected the food items, they may proceed to select the delivery option. There should be **a minimum of 3 delivery options** – Direct (<30 mins), Standard (45 mins), Saver (60 mins). You may assume that the user's delivery address is either <1km or 5km or 10km from the restaurant.
- e) **Order summary** – once the delivery option has been selected, the customer may proceed to **view the order summary** which includes the **subtotal, delivery fee and total amount to pay**.
- f) **Payment** – the customer should then proceed to **make payment** for the food order. There should be **a minimum of 3 payment methods** (eg. Credit card, e-wallet, cash on delivery, etc.).
- g) **Confirmation** – once the customer enters/selects the payment method and confirms the placement of the order, your application should provide a **confirmation** of the order which includes a **unique order id**, summary of the food order and the details of the **rider** who has been assigned to deliver your order.
- h) **Reorder** – the application should allow the user to **view a past order and to automatically order the same items**.
- i) **Additional functions** – include **2 relevant additional functions** (not described in this assignment) that will further demonstrate your ability to use object-oriented programming (OOP) concepts
- j) **Code Standard**
- You must demonstrate your ability to use as many OOP concepts as possible for your project. **Use classes and objects, composition, inheritance, and polymorphism where necessary**. At least one **static** function must also be applied in your program.
  - Your project should have **one main.cpp source file, multiple header files, and text file(s)**.
  - All data should be saved into text files **preferably in CSV format**.
  - Use **exception handling** to handle possible **input errors**.
  - Use suitable **STLs**.

## Part II: Project Documentation (30 Marks)

Create a project documentation that covers the following details:

- a) A set of UML diagrams that shows the functionalities, relationship between classes, and the workflow of your food ordering application. Use appropriate UML tools and diagramming techniques.
- b) Explain why the STLs you used to develop the food ordering application is suitable based on this project context.
- c) Explain the rationale of using inheritance and polymorphism you implemented in your project. For example, you may clarify why a class should inherit from another class(es) and how you enabled polymorphism in your program codes.

Assignment Project Quiz Exam Essay Help  
WeChat: cestbon-688  
Email: accoder-overseas@163.com

# APPENDIX 1

## MARKING RUBRICS

Component Title	Part 1: Food Ordering Application					Percentage (%)	35
Criteria	Score and Descriptors					Weight (%)	Marks
	Excellent (5)	Good (4)	Average (3)	Need Improvement (2)	Poor (1)		
Classes and objects	Classes are well organised and implemented with multiple header files and if necessary multiple CPP files.	Classes and objects are fully implemented with the separation of class specification implementation.	Classes and objects are fully implemented.	Partial Implementation of classes and objects.	Classes and object were not implemented or is not used at all.	15	
Inheritance	Excellent implementation of Inheritance with base classes.	Adequate implementation of inheritance.	Minimal inheritance implemented.	Partial inheritance implemented.	No inheritance seen in the code.	10	
Polymorphism	Excellent implementation of polymorphism with abstract classes.	Adequate implementation of polymorphism that covers function overloading and operator overloading.	Minimal implementation of polymorphism with some function and operator overloading.	Partial implementation of polymorphism with some function and operator overloading.	No polymorphism implemented.	10	
STL	Excellent STL implementation with self-developed template libraries.	Adequate use of STL.	Minimal use of STL.	Partial or improper use of STL	No STL used.	10	
Files	Excellent implementation of file usage for data storage and retrieval. This includes searching and retrieving algorithm.	Adequate use of files for storage. Data and configuration information is also stored and retrieved from files.	Minimal use of files for storage. All required data are stored and retrieved from files.	Partial use of files for storage. Not all data are stored / retrieved from file	No file storage implemented.	5	
Exception handling	Excellent exception	Adequate exception	Minimal exception	Partial exception	No exception handling	5	

	handling with all software errors caught and handled appropriately.	handling by handling all important errors as well as recovering from some of those errors.	handling where important errors are handled by printing out proper error statements.	handling where error is acknowledged and program exits.	implemented.		
Quality of implementation	Excellent coding format and standards. External files are named appropriately. Development uses project files for multi-file source code.	Program runs very well with good UI. Code conforms to coding standards and format.	Program runs perfectly with simple UI.	Program can compile and run but crashes.	Program cannot compile and/or run.	15	
TOTAL						70	

Component Title	Part II: Project Documentation					Percentage (%)	15	
Criteria	Score and Descriptors					Weight (%)	Marks	
	Excellent (5)	Good (4)	Average (3)	Need Improvement (2)	Poor (1)			
UML Diagrams	Precise and accurate usage of the diagram.	A few minor mistakes were found in the diagram (e.g., wrong class name).	Some noticeable mistakes (e.g., inaccurate class relationships)	Not all the classes are included in the diagram.	The diagram is confusing, and the format is not standardised. or no attempt	10		
STL Explanation	The explanations are clear and concise to address the question accurately	The explanations are good but missing a few essential points.	The explanations are acceptable but with a few inaccurate statements.	The explanations are vague and contain several inaccurate statements.	The explanations are almost irrelevant or wrong or no attempt	10		
Inheritance and polymorphism explanation	The explanations are clear and concise to address the question accurately	The explanations are good but missing a few essential points.	The explanations are acceptable but with a few inaccurate statements.	The explanations are vague and contain several inaccurate statements	The explanations are almost irrelevant or wrong or no attempt	10		
TOTAL						30		