MODULE NAME:	MODULE CODE:
PROGRAMMING 2A	PROG6221

ASSESSMENT TYPE: POE (PAPER & MEMORANDUM)

TOTAL MARK ALLOCATION: 300 MARKS

TOTAL HOURS: A minimum of 35 HOURS is suggested to complete this assessment.

By submitting this assignment, you acknowledge that you have read and understood all the rules as per the terms in the registration contract, in particular the assignment and assessment rules in The IIE Assessment Strategy and Policy (IIE009), the intellectual integrity and plagiarism rules in the Intellectual Integrity and Property Rights Policy (IIE023), as well as any rules and regulations published in the student portal.

INSTRUCTIONS:

1. ***No material may be copied from original sources, even if referenced correctly, unless it is a direct quote indicated with quotation marks. No more than 10% of the assignment may consist of direct quotes.***
2. ***Make a copy of your assignment before handing it in.***
3. *Assignments must be typed unless otherwise specified.*
4. *Begin each section on a new page.*
5. *Follow all instructions on the PoE cover sheet.*
6. *This is an individual assignment.*

ACADEMIC HONESTY DECLARATION

Please complete the Academic Honesty Declaration below.
Please note that your assessment will not be marked, and you will receive 0% if you have not completed ALL aspects of this declaration.

Declaration

	SIGN
I have read the assessment rules provided in this declaration.	
This assessment is my own work.	
I have not copied any other student's work in this assessment.	
I have not uploaded the assessment question to any website or App offering assessment assistance.	
I have not downloaded my assessment response from a website.	
I have not used any AI tool without reviewing, re-writing, and re-working this information, and referencing any AI tools in my work.	
I have not shared this assessment with any other student.	
I have not presented the work of published sources as my own work.	
I have correctly cited all my sources of information.	
My referencing is technically correct, consistent, and congruent.	
I have acted in an academically honest way in this assessment.	

Referencing Rubric

Providing evidence based on valid and referenced academic sources is a fundamental educational principle and the cornerstone of high-quality academic work. Hence, The IIE considers it essential to develop the referencing skills of our students in our commitment to achieve high academic standards. Part of achieving these high standards is referencing in a way that is consistent, technically correct and congruent. This is not plagiarism, which is handled differently.

Poor quality formatting in your referencing will result in a penalty **of a maximum of ten percent being deducted from the percentage awarded**, according to the following guidelines. Please note, however, that **evidence of plagiarism in the form of copied or uncited work (not referenced), absent reference lists, or exceptionally poor referencing, may result in action being taken in accordance with The IIE's Intellectual Integrity Policy (0023)**.

Markers are required to provide feedback to students by indicating **(circling/underlining) the information that best describes the student's work.**

Minor technical referencing errors: 5% deduction from the overall percentage – the student's work contains **five or more errors** listed in the minor errors column in the table below.

Major technical referencing errors: 10% deduction from the overall percentage – the student's work contains **five or more errors** listed in the major errors column in the table below**.**

If both minor and major errors are indicated, then 10% only (and not 5% or 15%) is deducted from the overall percentage. The examples provided below are not exhaustive but are provided to illustrate the error

Required: **Technically correct referencing style**	**Minor errors in technical correctness of referencing style** **Deduct 5% from percentage awarded**	**Major errors in technical correctness of referencing style** **Deduct 10% from percentage awarded**
Consistency • The same referencing format has been used for all in-text references and in the bibliography/reference list.	Minor inconsistencies. • The referencing style is generally consistent, but there are one or two changes in the format of in-text referencing and/or in the bibliography. • For example, page numbers for direct quotes (in-text) have been provided for one source, but not in another instance. Two book chapters (bibliography) have been referenced in the bibliography in two different formats.	Major inconsistencies. • Poor and inconsistent referencing style used in-text and/or in the bibliography/ reference list. • Multiple formats for the same type of referencing have been used. • For example, the format for direct quotes (in-text) and/or book chapters (bibliography/ reference list) is different across multiple instances.
Technical correctness • Referencing format is technically correct throughout the submission. • The correct referencing format for the **module's discipline** has been used, i.e., either **APA,** OR **Harvard** OR **Law.** • Position of the reference: a reference is directly associated with every concept or idea. • For example, quotation marks, page numbers, years, etc. are applied correctly, sources in the bibliography/reference list are correctly presented.	**Generally, technically correct with some minor errors.** • The correct referencing format has been consistently used, but there are one or two errors. • Concepts and ideas are typically referenced, but a reference is missing from one small section of the work. • Position of the references: references are only given at the beginning or end of every paragraph. • For example, the student has incorrectly presented direct quotes (in-text) and/or book chapters (bibliography/reference list).	**Technically incorrect.** • The referencing format is incorrect. • Concepts and ideas are typically referenced, but a reference is missing from small sections of the work. • Position of the references: references are only given at the beginning or end of large sections of work. • For example, incorrect author information is provided, no year of publication is provided, quotation marks and/or page numbers for direct quotes missing, page numbers are provided for paraphrased material, the incorrect punctuation is used (in-text); the bibliography/reference list is not in alphabetical order, the incorrect format for a book chapter/journal article is used, information is missing e.g. no place of publication had been provided (bibliography); repeated sources on the reference list.
Congruence between in-text referencing and bibliography/ reference list • All sources are accurately reflected and are all accurately included in the bibliography/ reference list.	**Generally, congruence between the in-text referencing and the bibliography/ reference list with one or two errors.** • There is largely a match between the sources presented in-text and the bibliography. • For example, a source appears in the text, but not in the bibliography/ reference list or vice versa.	**A lack of congruence between the in-text referencing and the bibliography.** • No relationship/several incongruencies between the in-text referencing and the bibliography/reference list. • For example, sources are included in-text, but not in the bibliography and vice versa, a link, rather than the actual reference is provided in the bibliography.
In summary: the recording of references is accurate and complete.	In summary, at least **80%** of the sources are correctly reflected and included in a reference list.	In summary, at least **60%** of the sources are incorrectly reflected and/or not included in reference list.

Overall Feedback about the consistency, technical correctness and congruence between in-text referencing and bibliography:

.....

Background

Sanele was invited to Lindiwe's birthday party.
He would have attended anyway since she is a good friend of his. But when he saw the party would be at her parents' house and the instruction was to "come hungry," he was intrigued.
So, he dutifully skipped lunch the day of the party.

As he walked up to the house, the smell of barbecue started getting stronger. He was still standing with his eyes closed and a silly grin on his face, inhaling as deeply as he could when she opened the door.
Her melodious laugh brought him suddenly back to reality.
After an embarrassing moment, he remembered to wish her a happy birthday.

It turns out Lindiwe's parents have some excellent skills when it comes to cooking. And some secret family recipes, too, it is said.
Sanele was glad that he followed the instruction to come hungry! Chicken, beef, and a leg of lamb were on the braai, all cooked to perfection. And don't forget about the roasted mielies.
There was a big pot of pap with a very flavourful chakalaka sauce next to it.
There were salads, roast vegetables, and sweet potatoes. And freshly baked bread straight from the oven. Sanele was in heaven.

Just when he thought the day couldn't get any better, it was time for dessert. There was malva pudding with custard and chocolate pudding with chocolate ice cream. When Lindiwe's dad spotted Sanele stuck choosing between the two, he casually suggested, "Why not have both?"

That day, Sanele decided he needed to learn how to cook fantastic food like that.
If a lawyer and a doctor can do this in their free time, so can he.
In this portfolio of evidence, you will develop a recipe app to start him on his journey.

Instructions

This portfolio of evidence consists of three parts–two parts submitted during the semester and a final submission at the end of the semester.
The parts build on one another, so keep a copy of your work saf**e**.

In the first part, you will create a command-line application that allows the user to enter and store the ingredients and steps for one recipe. In the second part, you will extend it to support multiple recipes and include nutritional information. In the final submission, you will change the user interface to a more user-friendly graphical one.

The requirements of real software projects frequently change, often in quite unexpected ways. Here, you have the benefit of knowing what all the requirements will be in advance.
So, make use of the opportunity. **Reading all three parts** before starting with the first one will minimise any reworking for later parts.

The **rubrics** that will be used to mark your submissions appear at the end of this document.
Please pay attention to the weighting of items in the rubrics.

Note that marks will be awarded for **running functional software,** not just source code.
So, ensure that your source code **compiles,** and that the **readme** file contains enough information about running the software.

Part 1 — Object-Oriented Programming (Marks: 100)

Learning Units 1 and 2

At the end of this specific part, students should be able to:

- Write a console programme that requires user input.
- Apply string manipulation to solve a programming problem.
- Use automatic properties to solve a programming problem.

For this portfolio of evidence, you must store your source code in a **GitHub repository**.

Make regular **commits** with descriptive commit **comments**. Marks will be awarded for this (5%), but more importantly, it will help to keep your code safe.

Using **C#** and **Visual Studio**, design and implement a standalone **command line application** that fulfils the following requirements:

1. The user shall be able to **enter** the details for a single **recipe**:
   a. The **number** of ingredients.
   b. For each **ingredient**: the name, quantity, and unit of measurement. For example, one tablespoon of sugar.
   c. The **number** of steps.
   d. For each **step**: a description of what the user should do.
2. The software shall display the **full recipe**, including the ingredients and steps, in a neat format to the user.
3. The user shall be able to request that the recipe is **scaled** by a factor of 0.5 (half), 2 (double), or 3 (triple). All the **ingredient quantities** shall be changed accordingly when the recipe is displayed. For example, one tablespoon of sugar will become two tablespoons of sugar if the factor is 2.
4. The user can request that the **quantities** be **reset** to the **original values**.
5. The user shall be able to **clear all the data** to enter a new recipe.
6. The software shall **not persist** the user data between runs. The data shall only be stored in memory while the software is running.

Non-functional requirements:

1. You are required to use internationally acceptable **coding standards**. Include comprehensive comments explaining variable names, methods, and the logic of programming code.
2. You are required to use **classes**.
3. Store the **ingredients** and steps in **arrays**.

When you are ready to submit Part 1, create a **tag** called **Part1** in your GitHub repository.

Tip: Make sure your lecturer has access to your repository.

Submit the following items for this part:

1. A **zip file** containing the complete **source code**, including the Visual Studio project files.

2. A **readme file** containing:
a. instructions for how to compile and run the software; and
b. a link to your GitHub repository.
3. A **screenshot** of your **GitHub repository** showing the commit history up to the Part1 tag.

Part 2 — Advanced C# Features (Marks: 100)

Learning Units 1 to 3

At the end of this specific part, students should be able to:

- *Use a generic collection to solve a programming problem.*
- *Use delegates to solve a programming problem.*

You will continue working on the application created in Part 1. **Implement** the **feedback** provided by your lecturer on Part 1 before continuing with Part 2. Marks will be awarded for this (10%).

The application must still perform all the functions from Part 1, with the following features added:

1. The user shall be able to enter an **unlimited number of recipes**.
2. The user shall be able to enter a **name** for each **recipe**.
3. The software shall **display** a **list** of all the **recipes** to the user in **alphabetical order** by **name**.
4. The user can choose which **recipe to display** from the list.
5. For each **ingredient**, the user shall additionally be able to enter:
a. The number of **calories, and**
b. The **food group** that the ingredient belongs to.
6. The software shall calculate and display the **total calories** of all the ingredients in a **recipe**.
7. The software shall notify the user when the **total calories** of a recipe **exceed 300**.

Read more about food groups here: https://sweetlife.org.za/what-are-the-different-food-groups-a-simple-explanation/ [Accessed on 14 February 2024].

Non-functional requirements:

8. You are required to use internationally acceptable **coding standards**. Include comprehensive comments explaining variable names, methods, and the logic of programming code.

9. You are required to use **classes**.
10. You must use generic collections to store the recipes, ingredients, and steps, and no longer arrays.
11. You are required to use a **delegate** to notify the user when a recipe exceeds 300 calories.
12. You are required to create a **unit test** to test the **total calorie calculation**.

When you are ready to submit this part, create a **tag** called **Part2** in your GitHub repository.

Submit the following items for this part:

1. A **zip file** containing the complete **source code**, including the Visual Studio project files.
2. A **readme file** containing:

a. instructions for how to compile and run the software;
b. a link to your GitHub repository; and
c. a brief description (100 to 200 words) of what you changed based on your lecturer's feedback.

3. A **screenshot** of your **GitHub repository** showing the commit history up to the Part 2 tag.

Portfolio of Evidence (PoE) — Windows Presentation Foundation (Marks: 100)

All learning units

At the end of this specific part, students should be able to:

- *Use the Extensible Application Markup Language to create graphical user interfaces.*
- *Use controls to create a graphical user interface.*
- *Use graphics rendering services to display graphical views of data.*
- *Use styles in a user interface.*

You will continue working on the application created in Part 2. **Implement** the **feedback** provided by your lecturer on Part 2 before continuing with the final PoE submission. Marks will be awarded for this (10%).

For this part, you are required to update your application to have a graphical user interface (GUI) built using *either* Windows Presentation Foundation (**WPF**) *or* Universal Windows Platform (**UWP**). Note that UWP will require additional research, so choose wisely.

All the same functionalities must be available in the new user interface that was in the command line application from Part 2 (just presented in a **more user-friendly way**), with your **choice** of **one** of the following features added:

1. The user shall be able to **filter the list of recipes** by:
   a. entering the name of an **ingredient** that must be in the recipe,
   b. choosing a **food group** that must be in the recipe, or
   c. selecting a **maximum** number of **calories**.

or

2. The user can choose **multiple recipes** to include in a **menu**. The software then displays a **pie chart** showing the **percentage** that each **food group** makes up of the total menu.

When ready to submit this part, create a **tag** called **PoE** in your GitHub repository.

Submit the following items for this part:

1. A **zip file** containing the full **source code**, including the Visual Studio project files.
2. A **readme file** containing:
   a. instructions for how to compile and run the software;
   b. a link to your GitHub repository; and
   c. a brief description (100 to 200 words) of what you changed based on your lecturer's feedback.
3. A **screenshot** of your **GitHub repository** showing the commit history up to the POE tag.
4. A short **user manual** (no more than 2000 words), including **screenshots**, which explains how to use the app. You may use any application of your choice to create the user manual, but the file you submit must be a **.PDF export** of the document.

Assessment Sheet (Marking Rubric)

Please note: Tear off this section and **attach** it to your work when you submit it/ If this is an online submission, then this information needs to be included in the online submission.

MODULE NAME:	MODULE CODE:
PROGRAMMING 2A	PROG6221

STUDENT NAME:
STUDENT NUMBER:

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PART 1					
Repository management: GitHub repository created and used to store code. **[5 Marks]**	• No evidence was submitted of a GitHub repository. • A repository was created, but no commits were made. • A repository was created with only a single commit. • Commit comments does	• Evidence of repository usage was submitted. • At least 5 commits were made with somewhat descriptive commit comments. • A tag called Part1 was created.	• Evidence of repository usage was submitted. • At least 10 commits were made with clear commit comments. • A tag called Part1 was created.	• Evidence of repository usage was submitted. • At least 15 commits were made with extensive commit comments. • A tag called Part1 was created.	

Formatted Table

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PART 1					
	not provide any information.				
	0 – 2 Marks	**3 Marks**	**4 Marks**	**5 Marks**	
App Functionality: The user can enter ingredients and steps, and the data is stored in memory. **[15 Marks]**	• The program does not compile. • The ingredients and steps cannot be entered. • The app crashes regardless of what the user enters. • The ingredients and steps can be entered but are not stored in memory.	• All the values can be entered, but no error handling has been implemented. • The entered values are stored in memory.	• All the values can be entered, but error handling could be improved. • The entered values are stored in memory.	• All values can be entered, and good error handling is implemented. • The entered values are stored in memory.	
	0 – 7 Marks	**8 – 9 Marks**	**10 – 11 Marks**	**12 – 15 Marks**	
App Functionality: The entered recipe is displayed in a neat format to the user. **[10 Marks]**	• The program does not compile. • The recipe is not displayed at all. • The recipe is displayed, but the data is incorrect.	• The ingredients and steps are displayed, but the layout can be significantly improved.	• The ingredients and steps are displayed, with some improvements that can be made to the layout.	• The recipe is displayed to the user in a neat format, with the steps numbered	

Formatted Table

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PART 1					
				and ingredients neatly laid out. • The app uses advanced features such as coloured text in the display.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
App Functionality: The recipe can be scaled with all ingredients scaled accordingly. **[15 Marks]**	• The program does not compile. • The recipe cannot be scaled at all. • The recipe can be scaled, but only some of the ingredients are affected. • The values are calculated but not displayed.	• All the required factors can scale the recipe. • The recipe is displayed with the scaled values.	• All the required factors can scale the recipe. • The recipe display adapts well to the changing values.	• All the required factors can scale the recipe. • Units of measurement are changed correctly when scaling. For example, 8 tablespoons multiplied by 2 becomes 1 cup.	
	0 – 7 Marks	**8 – 9 Marks**	**10 – 11 Marks**	**12 – 15 Marks**	

Formatted Table

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PART 1					
App Functionality: The recipe scale can be reset back to the original values. **[5 Marks]**	• The program does not compile. • The recipe cannot be reset back to its original values. • The recipe is not displayed again after resetting to the original values.	• The recipe can be reset to its original values. • The recipe is displayed after being reset.	• The recipe can be reset to its original values. • The recipe is displayed after being reset. • The recipe display adapts well to the changing values.	• All the required factors can scale the recipe. • Units of measurement are changed back correctly, resetting if they were changed.	
	0 – 2 Marks	**3 Marks**	**4 Marks**	**5 Marks**	
App Functionality: The data can be cleared, and a new recipe entered. **[10 Marks]**	• The program does not compile. • The data cannot be cleared. • The data can only partially be cleared.	• The data can be cleared but entering new data could be handled better. • The user is not asked to confirm before clearing.	• The data can be cleared. • The user is not asked to confirm before clearing. • New data entry is handled well.	• The data can be cleared. • The user is asked to confirm before clearing. • New data entry works well.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
Application Structure: The application makes use of classes in a logical way. **[10 Marks]**	• The class structure is completely illogical and confusing.	• The class structure is somewhat logical, with some errors.	• The class structure is mostly logical, with a few minor errors.	• The class structure is logical and easy to follow.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

Formatted Table

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PART 1					
Application Structure: The ingredients and steps are stored in an array. **[10 Marks]**	• Ingredients and steps are not stored in arrays. • The app crashes due to array size problems. **0 – 4 Marks**	• Ingredients and steps are stored in arrays. • Management of the array size works most of the time. **5 Marks**	• Ingredients and steps are stored in arrays. • The array size can be managed a little better. **6 – 7 Marks**	• Ingredients and steps are stored in arrays. • The array size is managed well. **8 – 10 Marks**	
Coding Standards: Code is well structured and documented. **[10 Marks]**	• The code is all in one file with no comments. **0 – 4 Marks**	• The code is structured somewhat well, with some comments. **5 Marks**	• The code is well structured with minor mistakes and mostly commented. **6 – 7 Marks**	• The code is well structured, with good comments explaining the logic. **8 – 10 Marks**	
Documenta-tion: Readme file provides enough information to run the app. **[10 Marks]**	• No readme file is included, or the readme file doesn't provide any helpful information about running the application. • The readme file contains information about running the	• The readme file presents some information about running the app but could be more detailed.	• The readme file presents most of the information about running the app but could be more detailed.	• An excellent readme file is included that explains all the required details about running the app.	

Formatted Table

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PART 1					
	app, but it is hard to understand or doesn't work.				
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PART 2					
Updates: The updates according to the readme file are correctly implemented. **[10 Marks]**	• No readme file was submitted. • No updates were listed in the readme file. • Most of the updates listed in the readme file were not implemented.	• Some of the updates in the readme file were well implemented.	• The updates described in the readme file were mostly well implemented.	• The updates described in the readme file were all well implemented.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
Unit test: A unit test was implemented to test the calorie calculation. **[5 Marks]**	• No unit test was submitted. • The unit test code does not compile. • The unit test code doesn't test the calorie calculation.	• The unit test covers the most basic calorie calculation.	• The unit test covers some additional possible scenarios with the calorie calculation.	• The unit test extensively covers every possible scenario with the calorie calculation.	

Formatted Table

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PART 1					
	0 – 2 Marks	**3 Marks**	**4 Marks**	**5 Marks**	
App Functionality: The user can enter unlimited recipes each with a name. **[10 Marks]**	• The program does not compile. • No recipes can be entered. • Only one recipe can be entered. • More recipes can be entered, but only one is stored in memory.	• An unlimited number of recipes can be entered, each with a name. • The process of entering more recipes is not obvious.	• An unlimited number of recipes can be entered, each with a name. • The flow can be improved to make entering more recipes easier.	• An unlimited number of recipes can be entered, each with a name. • The process of adding more recipes is easy to understand.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
App Functionality: The app displays the list of recipes in alphabetical order. **[10 Marks]**	• The program does not compile. • No list of recipes is displayed. • The list of recipes is displayed but is not sorted in alphabetical order.	• The list of recipes is displayed in alphabetical order, but the display could be improved significantly.	• The list of recipes is displayed in alphabetical order, but the display can be somewhat improved.	• The list of recipes is displayed in alphabetical order, and the display is excellently done. • The app makes use of advanced features such as coloured text.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

Formatted Table

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
			PART 1		
App Functionality: The user can enter calories and a food group for each ingredient. **[5 Marks]**	• The program does not compile. • Calories and food group cannot be entered. • Calories and food group can be entered but are not stored in memory.	• Calories and food group can be entered and stored in memory.	• Calories and food group can be entered and stored in memory. • An explanation is shown to the user of what these values mean.	• Calories and food group can be entered and stored in memory. • An explanation is shown to the user of what these values mean. • The user can select the food group from different options.	
	0 – 2 Marks	**3 Marks**	**4 Marks**	**5 Marks**	
App Functionality: The total calories of a recipe is calculated and displayed. **[10 Marks]**	• The program does not compile. • The total calories of a recipe are not calculated. • The total calories of a recipe are calculated but not displayed.	• The total calories of a recipe are correctly calculated and displayed. • The display could be improved.	• The total calories of a recipe are correctly calculated and displayed. • An explanation is included of what calories are.	• The total calories of a recipe are correctly calculated and displayed. • An explanation is included that is specific to certain ranges of calories.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

Formatted Table

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PART 1					
App Functionality: The user is alerted when the calories of a recipe exceed 300. **[10 Marks]**	• The program does not compile. • The total calories are not calculated. • There is no alert when the calories exceed 300. • The alert was implemented but didn't work as expected.	• An alert is displayed when the calories exceed 300. • No additional information is provided.	• An alert is displayed when the calories exceed 300. • General information about calories is included in the alert.	• An alert is displayed when the calories exceed 300. • Information relevant to the number of calories is displayed to the user as part of the alert.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
Application Structure: The recipes, ingredients and steps are stored in generic collections. **[10 Marks]**	• The program does not compile. • None of the data is stored in a generic collection. • Only one of the data types is stored in a generic collection. • Data is stored in non-generic collections.	• Recipes and ingredients are stored in generic collections, but not steps.	• The recipes, ingredients and steps are all stored in generic collections.	• The recipes, ingredients and steps are all stored in generic collections. • The code makes good use of all the relevant features of generic collections.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

Formatted Table

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PART 1					
Application Structure: The 300-calorie notification is done using a delegate. **[10 Marks]**	• The 300-calorie notification is not implemented at all or does not work at runtime. • The 300-calorie notification is implemented using something other than a delegate.	• The 300-calorie notification is implemented using a delegate that will work some of the time.	• The 300-calorie notification is implemented using a delegate. • The program flow doesn't continue naturally after the notification.	• The 300-calorie notification is excellently implemented using a delegate. • The program flow continues smoothly after the notification.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
Coding Standards: Code is well structured and documented. **[10 Marks]**	• The code is all in one file with no comments.	• The code is structured somewhat well, with some comments.	• The code is well structured with minor mistakes and mostly commented.	• The code is well structured, with good comments explaining the logic.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

Formatted Table

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
		PART 2			
Updates: The updates according to the readme file are correctly implemented. [10 Marks]	• No readme file was submitted. • No updates were listed in the readme file. • Most of the updates listed in the readme file were not implemented.	• Some of the updates in the readme file were well implemented.	• The updates described in the readme file were mostly well implemented.	• The updates described in the readme file were all well implemented.	
	0 – 4 Marks	5 Marks	6 – 7 Marks	8 – 10 Marks	
Unit test: A unit test was implemented to test the calorie calculation. [5 Marks]	• No unit test was submitted. • The unit test code does not compile. • The unit test code doesn't test the calorie calculation.	• The unit test covers the most basic calorie calculation.	• The unit test covers some additional possible scenarios with the calorie calculation.	• The unit test extensively covers every possible scenario with the calorie calculation.	
	0 – 2 Marks	3 Marks	4 Marks	5 Marks	

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
		PART 2			
App Functionality: The user can enter unlimited recipes each with a name. [10 Marks]	• The program does not compile. • No recipes can be entered. • Only one recipe can be entered. • More recipes can be entered, but only one is stored in memory.	• An unlimited number of recipes can be entered, each with a name. • The process of entering more recipes is not obvious.	• An unlimited number of recipes can be entered, each with a name. • The flow can be improved to make entering more recipes easier.	• An unlimited number of recipes can be entered, each with a name. • The process of adding more recipes is easy to understand.	
	0 – 4 Marks	5 Marks	6 – 7 Marks	8 – 10 Marks	
App Functionality: The app displays the list of recipes in alphabetical order. [10 Marks]	• The program does not compile. • No list of recipes is displayed. • The list of recipes is displayed but is not sorted in alphabetical order.	• The list of recipes is displayed in alphabetical order, but the display could be improved significantly.	• The list of recipes is displayed in alphabetical order, but the display can be somewhat improved.	• The list of recipes is displayed in alphabetical order, and the display is excellently done. • The app makes use of advanced features such as coloured text.	
	0 – 4 Marks	5 Marks	6 – 7 Marks	8 – 10 Marks	

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
			PART 2		
App Functionality: The user can enter calories and a food group for each ingredient. [5 Marks]	• The program does not compile. • Calories and food group cannot be entered. • Calories and food group can be entered but are not stored in memory.	• Calories and food group can be entered and stored in memory.	• Calories and food group can be entered and stored in memory. • An explanation is shown to the user of what these values mean.	• Calories and food group can be entered and stored in memory. • An explanation is shown to the user of what these values mean. • The user can select the food group from different options.	
	0 – 2 Marks	3 Marks	4 Marks	5 Marks	
App Functionality: The total calories of a recipe is calculated and displayed. [10 Marks]	• The program does not compile. • The total calories of a recipe are not calculated. • The total calories of a recipe are calculated but not displayed.	• The total calories of a recipe are correctly calculated and displayed. • The display could be improved.	• The total calories of a recipe are correctly calculated and displayed. • An explanation is included of what calories are.	• The total calories of a recipe are correctly calculated and displayed. • An explanation is included that is specific to certain ranges of calories.	
	0 – 4 Marks	5 Marks	6 – 7 Marks	8 – 10 Marks	

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
			PART 2		
App Functionality: The user is alerted when the calories of a recipe exceed 300. [10 Marks]	• The program does not compile. • The total calories are not calculated. • There is no alert when the calories exceed 300. • The alert was implemented but didn't work as expected.	• An alert is displayed when the calories exceed 300. • No additional information is provided.	• An alert is displayed when the calories exceed 300. • General information about calories is included in the alert.	• An alert is displayed when the calories exceed 300. • Information relevant to the number of calories is displayed to the user as part of the alert.	
	0 – 4 Marks	5 Marks	6 – 7 Marks	8 – 10 Marks	
Application Structure: The recipes, ingredients and steps are stored in generic collections. [10 Marks]	• The program does not compile. • None of the data is stored in a generic collection. • Only one of the data types is stored in a generic collection. • Data is stored in non-generic collections.	• Recipes and ingredients are stored in generic collections, but not steps.	• The recipes, ingredients and steps are all stored in generic collections.	• The recipes, ingredients and steps are all stored in generic collections. • The code makes good use of all the relevant features of generic collections.	
	0 – 4 Marks	5 Marks	6 – 7 Marks	8 – 10 Marks	

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
			PART 2		
Application Structure: The 300-calorie notification is done using a delegate. [10 Marks]	• The 300-calorie notification is not implemented at all or does not work at runtime. • The 300-calorie notification is implemented using something other than a delegate.	• The 300-calorie notification is implemented using a delegate that will work some of the time.	• The 300-calorie notification is implemented using a delegate. • The program flow doesn't continue naturally after the notification.	• The 300-calorie notification is excellently implemented using a delegate. • The program flow continues smoothly after the notification.	
	0 – 4 Marks	5 Marks	6 – 7 Marks	8 – 10 Marks	
Coding Standards: Code is well structured and documented. [10 Marks]	• The code is all in one file with no comments.	• The code is structured somewhat well, with some comments.	• The code is well structured with minor mistakes and mostly commented.	• The code is well structured, with good comments explaining the logic.	
	0 – 4 Marks	5 Marks	6 – 7 Marks	8 – 10 Marks	

Documentation: The readme file provides enough information to run the app. **[10 Marks]**	• No readme file is included, or the readme file doesn't provide any helpful information about running the application. • The readme file contains information about running the app, but it is hard to understand or doesn't work.	• The readme file presents some information about running the app but could be more detailed.	• The readme file presents most of the information about running the app but could be more detailed.	• An excellent readme file is included that explains all the required details about running the app.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PORTFOLIO OF EVIDENCE (POE)					
Updates: The updates according to the readme file are correctly implemented. **[10 Marks]**	• No readme file was submitted. • No updates were listed in the readme file. • Most of the updates listed in the readme file were not implemented.	• Some of the updates in the readme file were well implemented.	• The updates described in the readme file were mostly well-implemented.	• The updates described in the readme file were all well implemented.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
App Functionality: The user can enter unlimited recipes with a name, ingredients, and steps. **[10 Marks]**	• The program does not compile. • The user can enter only one recipe. • The app crashes when the user tries to enter more than one recipe.	• The program allows the user to enter multiple recipes, but the process is not easy to use.	• The program allows the user to enter multiple recipes. • The process could be a little easier.	• The program allows the user to enter multiple recipes. • The user can easily know how to enter more recipes. • The program makes entering a recipe easy by allowing selections where possible instead of typing.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PORTFOLIO OF EVIDENCE (POE)					
App Functionality: The user can select a recipe to display from an alphabetical list of all the recipes. **[10 Marks]**	• The program does not compile. • No list of recipes is displayed. • The list of recipes is not alphabetical.	• A list of recipes is displayed with only the recipe name, in alphabetical order.	• The recipe list is displayed with some additional information besides the recipe name.	• The recipe list is displayed with a range of useful values in addition to the name.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
App functionality: The app can display a recipe in a user-friendly format. **[10 Marks]**	• The program does not compile. • The user cannot view a recipe. • A recipe can be displayed, but the display is incomplete or hard to read.	• A recipe can be displayed with the ingredients and steps. • No additional formatting or information is displayed.	• A recipe can be displayed with the ingredients and steps. • The steps are clearly numbered. • Some formatting is applied.	• A recipe can be displayed with the ingredients and steps. • Steps are clearly displayed and can be ticked off as the user completes the step. • Excellent formatting is applied.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PORTFOLIO OF EVIDENCE (POE)					
App functionality: Selected feature (filter or menu pie chart) works correctly. **[20 Marks]**	• The program does not compile. • No additional feature was implemented. • The additional feature doesn't work at all. • The feature is only partially implemented.	• The new feature was implemented with only the most basic functionality working.	• The new feature was implemented with some minor errors.	• The new feature was implemented successfully. • There were no errors with the implementation.	
	0 – 9 Marks	**10 – 12 Marks**	**13 – 14 Marks**	**15 – 20 Marks**	
Usability: User interface is easy to use. **[10 Marks]**	• The user interface is confusing and illogical.	• The user interface can be used but is not very logical.	• The user interface is well implemented, with a few small usability problems.	• The user interface is excellently implemented and very easy to use.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	
Coding Standards: Code is well structured and documented. **[10 Marks]**	• The code is all in one file with no comments.	• The code is structured somewhat well, with some comments.	• The code is well structured with minor mistakes and mostly commented.	• The code is well structured, with good comments explaining the logic.	
	0 – 4 Marks	**5 Marks**	**6 – 7 Marks**	**8 – 10 Marks**	

Marking Criteria	Does not meet the required standard	Meets the required standard	Partially exceeds the required standard	Greatly exceeds the required standard	Feedback
PORTFOLIO OF EVIDENCE (POE)					
Documentation: The user manual is well structured with useful screenshots. **[15 Marks]**	• Not submitted or almost no detail. • Some information is included.	• Enough detail is included to use the software based on the manual. • More screenshots are needed.	• The user manual included with some missing screenshots.	• Complete user manual included with good use of screenshots.	
	0 – 7 Marks	**8 – 9 Marks**	**10 – 11 Marks**	**12 – 15 Marks**	
Documentation: The readme file provides enough information to run the app. **[5 Marks]**	• No readme file is included, or the readme file doesn't provide any helpful information about running the application. • The readme file contains information about running the app, but it is hard to understand or doesn't work.	• The readme file presents some information about running the app but could be more detailed.	• The readme file presents most of the information about running the app but could be more detailed.	• An excellent readme file is included that explains all the required details about running the app.	
	0 – 2 Marks	**3 Marks**	**4 Marks**	**5 Marks**	
	0 – 2 Marks	**3 Marks**	**4 Marks**	**5 Marks**	

Appendix A - PoE Marking Rubrics

Markers – Please note that the rubrics below must be used to evaluate students' responses to the relevant assignment questions. Please clearly indicate the mark you allocate for each rubric criterion to show how you reached the question total. Also, please provide constructive feedback to ensure students and moderators can follow your marking logic based on the rubric criteria.

The most important point is that many markers across different campuses will be marking. The rubric needs to promote the validity and reliability of their marking practices. In addition, there is a separate memorandum that provides additional marking guidance – please ensure you get this from your relevant campus administrator.

While reading the students' submissions, the marker should consider the goals of the part and the description of each qualitative level.

The marker should match the student's work to the appropriate qualitative level, highlighting areas of achievement or areas that were not adequately addressed.

Once the marker has considered all of this, the student should only be assigned a mark.

PART 1 (Marks: 100)

Basic skills covered (not necessarily assessed)

LU1 and LU2

The emphasis in Part 1 is on basic object-oriented programming in C#. And the user interface should be command line only.

The instructions indicate that the ingredients should be stored in an array since collections are only discussed in learning unit 3. However, if a student uses a collection, award the marks.

PART 2 (Marks: 100)

Basic skills covered (not necessarily assessed)

LU1 to LU3

The description of changes in the readme file should serve as a guide for awarding marks for the application updates based on feedback. Check whether the described changes are implemented in the application, but it should be useful. The focus here is to add the new functionality and use the advanced C# features from learning unit 3.

PORTFOLIO OF EVIDENCE (POE) (Marks: 100)

Basic skills covered (not necessarily assessed)

All learning units

The POE focuses on building a user-friendly user interface for the application using WPF (or UWP), with one new feature added.

[TOTAL MARKS: 100]