MODULE NAME:	MODULE CODE:
PROGRAMMING 2A	PROG6211
PROGRAMMING 2A	PROG6221

ASSESSMENT TYPE: POE (PAPER)

TOTAL MARK ALLOCATION: 100 MARKS

TOTAL HOURS: A minimum of 15 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 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. ***Any assignment with a similarity index of more than 25% will be scrutinised for plagiarism. Please make sure you attach a similarity report to your POE if required.***
3. ***Make a copy of your assignment before handing it in.***
4. *Assignments must be typed unless otherwise specified.*
5. *All work must be adequately and correctly referenced.*
6. *Begin each section on a new page.*
7. *Follow all instructions on the assignment cover sheet.*
8. *This is an individual assignment.*

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.

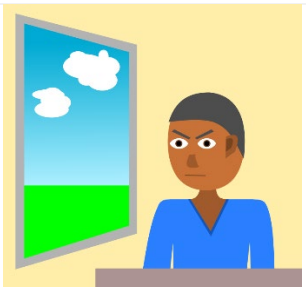
Required: **Technically correct referencing style**	**Minor errors in technical correctness of referencing style** **Deduct 5% from percentage awarded**	**Major errors in technical correctness of referencing style** **Deduct 10% from percentage awarded**
Consistency • The same referencing format has 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. 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:

.....

.....

Portfolio of Evidence (PoE) — Background



Siphiwe went home for a long weekend. You would think that he would come back rested and happy, having eaten his mother's amazing food all day long instead of his usual takeaways, in one of the most beautiful places in the country too. You would be wrong.

In class he was frequently staring out the window with an intense frown on his face. And between classes he didn't say much and didn't eat like his usual ravenous self either. When somebody jokingly told him that you can't live on water alone, his answer was serious. "No ... no, you cannot."

After a lot of prodding, he finally told you what had happened. His father had sat him down, and explained that once he graduates, he would no longer be receiving any money from the family. No car repayments, no entertainment allowance, no rent … Nothing. Siphiwe had never thought about what would happen after graduation. He was so focussed on his studies, that it had never even occurred to him that he might have to fend for himself in less than two years.

Siphiwe was worried that he might not be able to maintain his lifestyle if he had to pay for everything himself. What would he have to sacrifice to make ends meet? He didn't even have a girlfriend yet, and if he couldn't go out anymore, how would he ever find her? And even worse, if he did find her, how would he keep her without any money to spend on her? The myriad questions were spinning around in his head with no end in sight.

You pointed out to him that he needed some hard data. Without data he couldn't know if he even had financial trouble looming, let alone what he could do to improve his situation if he did.

In this Portfolio of Evidence, you will be creating an application that can be used for personal budget planning. That should help Siphiwe understand his potential problems much better.

Instructions

This Portfolio of Evidence consists of three parts – two tasks submitted during the semester and a final submission at the end of the semester. The tasks build on one another, so make sure that you **keep a copy of your work in a safe place**.

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

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, make sure that your source code **compiles,** and that the **readme** file contains enough information about running the software.

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

Learning Units: LU1 – 2

Assessment:

Assessment/Deliverable	Marks	Weight	Duration
Task 1	100	25%	15hrs

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 following **values**:
    a. Gross monthly income (before deductions).
    b. Estimated monthly tax deducted.
    c. Estimated monthly expenditures in each of the following categories:
        i. Groceries
        ii. Water and lights
        iii. Travel costs (including petrol)
        iv. Cell phone and telephone

v. Other expenses

2. The user shall be able to choose between **renting** accommodation or **buying a property**.
3. If the user selects to rent, the user shall be able to enter the monthly **rental amount**.
4. If the user selects to **buy a property**, the user shall be required to enter the following values for a home loan:
   a. Purchase price of property
   b. Total deposit
   c. Interest rate (percentage)
   d. Number of months to repay (between 240 and 360)
5. The software shall calculate the **monthly home loan repayment** for buying a property based on the values that the user entered. (See https://www.siyavula.com/read/maths/grade-10/finance-and-growth/09-finance-and-growth-03 for more information on how to calculate this).
6. If the monthly **home loan repayment** is more than a **third** of the user's gross monthly income, the software shall **alert** the user that approval of the home loan is unlikely.
7. The software shall **calculate** the **available monthly money** after all the specified deductions have been made.
8. 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** and **inheritance**. Create an abstract class Expense, from which HomeLoan, etc., can be derived.
3. Store the **expenses** in an **array**.

Submit the following items for this task:

1. **Source code**.
2. A **readme file** with instructions for how to compile and run the software.

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

Learning Units: LU1 – 3

Assessment:

Assessment/Deliverable	Marks	Weight	Duration
Task 2	100	30%	15hrs

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

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

1. The user shall be able to **choose** whether to **buy a vehicle**.
2. If the user selects to **buy a vehicle**, the user shall be required to enter the following values for vehicle financing:
    a. Model and make.
    b. Purchase price.
    c. Total deposit.
    d. Interest rate (percentage).
    e. Estimated insurance premium.
3. The software shall calculate the **total monthly cost** of buying the **car** (insurance plus loan repayment). Assume that all cars will be repaid over a period of **five years**.
4. The software shall notify the user when the **total expenses**, including loan repayments, **exceed 75%** of their **income**.
5. Display the **expenses** to the user in **descending order** by **value**.

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** and **inheritance**.
3. You are required to use a **generic collection** to store the expenses, and no longer an array.
4. You are required to use a **delegate** to notify the user when expenses exceed 75% of their income.

Submit the following items for this task:

1. **Source code**.
2. A **readme file** containing:
    a. Instructions for how to compile and run the software; and
    b. A brief description (100 to 200 words) of what you changed based on your lecturer's feedback.

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

Learning Unit: All

Assessment:

Assessment/Deliverable	Marks	Weight	Duration
Final Completed POE	100	35%	15hrs

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

For this task, 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 functionality must be available in the new user interface that was in the command line application from Task 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 choose to **save** up a specified amount by a certain date for a specified reason, e.g. save R100 000 for an honours degree over five years. Given the interest rate that will be earned on the savings, calculate how much the monthly saving should be to reach the goal.

OR

2. Display the monthly income, expenses and account balance (assume the balance starts at 0) as a graph over time, for a given period (e.g. five years).

Submit the following items for this task:

1. **Source code**.
2. A **readme file** containing:
    a. Instructions for how to compile and run the software; and
    b. A brief description (100 to 200 words) of what you changed based on your lecturer's feedback.
3. A short **user manual** (no more than 2 000 words) including **screenshots**, that explains how to use the app. You may use any application of your choice to create the user manual, but the file that you submit must be a **.PDF export** of the document.

Appendix A

Assessment Sheet (Marking Rubric)

Please note: Tear off this section and **attach** it to your work when you submit it.

MODULE NAME:	**MODULE CODE:**
PROGRAMMING 2A	**PROG6211**
PROGRAMMING 2A	**PROG6221**

STUDENT NAME:
STUDENT NUMBER:

RUBRIC 1	**Levels of Achievement**				**Feedback**
In order to be awarded full marks for these elements of Task 1, students need to have:	**Excellent**	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks possible)				
App Functionality: User can enter values for income, tax and expenditures.	**13—15** All values can be entered, and good error handling implemented.	**9—12** All the values can be entered, but error handling could be improved.	**4—8** Only some of the values can be entered or no error handling implemented.	**0—3** The values cannot be entered or the app crashes regardless of what the user enters.	

RUBRIC 1 [continued]	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of Task 1, students need to have:	**Excellent**	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks possible)				
App Functionality: Rental and home loan entry implemented correctly.	**13—15** All values can be entered, and good error handling implemented.	**9—12** All the values can be entered, but error handling could be improved.	**4—8** Only some of the values can be entered or no error handling implemented.	**0—3** The values cannot be entered or the app crashes regardless of what the user enters.	
App Functionality: Home loan calculations and warning implemented correctly.	**13—15** The home loan repayments are calculated correctly, and the alert is displayed in a user-friendly way.	**9—12** The home loan repayments are calculated correctly, but the alert is never displayed.	**4—8** The home loan repayments are calculated partially correctly, and the alert is never displayed.	**0—3** Home loan repayments not calculated, or the calculation is completely wrong.	
App Functionality: Available monthly money calculated correctly and displayed.	**13—15** Available monthly money calculated and displayed.	**9—12** Available money calculated but display could be improved.	**4—8** Available money partially correctly calculated or not displayed.	**0—3** Available money not calculated correctly and not displayed.	

RUBRIC 1 [continued]	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of Task 1, students need to have:	**Excellent**	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks possible)				
Application Structure: Application makes use of classes and inheritance in a logical way.	**9—10** Class structure is logical and easy to follow.	**7—8** Class structure is mostly logical with a few small errors.	**4—6** Class structure is somewhat logical with quite a few errors.	**0—3** Class structure is completely illogical and confusing.	
Application Structure: The expenses are stored in an array.	**9—10** Expenses are stored in an array and the array size is managed well.	**7—8** Most, but not all, the expenses are stored in the array.	**4—6** Only some of the expenses are stored in the array.	**0—3** Expenses are not stored in an array.	
Coding Standards: Code is well structured and documented.	**9—10** Code is well structured with good comments explaining logic.	**7—8** Code is well structured with minor mistakes and mostly commented.	**4—6** Code is structured somewhat well with little or no comments.	**0—3** Code is all in one file with no comments.	

RUBRIC 1 [continued]	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of Task 1, students need to have:	**Excellent**	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks possible)				
Documentation: Readme file provides enough information to run the app.	**9—10** An excellent readme file is included that explains all the required details about running the app.	**7—8** The readme file presents some information about running the app but could be more detailed.	**4—6** The readme file contains information about running the app, but it is hard to understand or doesn't work.	**0—3** No readme file is included, or the readme file doesn't provide any useful information about running the application.	
Other Marks: Advanced features not covered in class (Bonus Marks).	**5** Excellent use of features that go above and beyond what was covered in class.	**3—4** Good use of features that go above and beyond what was learned in class was used.	**1—2** Some features that go above and beyond what was learned in class was used.	**0** No features that go above and beyond what was learned in class was used.	
TASK 1 SUBTOTAL					**/100**

RUBRIC 2	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of Task 2, students need to have:	**Excellent**	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks possible)				
Updates: All the feedback provided on Task 1 has been implemented.	**9—10** Excellent implementation of all feedback provided.	**7—8** Most feedback was implemented.	**4—6** Some feedback was implemented.	**0—3** Little or no feedback was implemented.	
App Functionality: The user can successfully capture the details for a vehicle.	**16—20** All values can be entered, and good error handling implemented.	**10—15** All the values can be entered, but error handling could be improved.	**5—9** Only some of the values can be entered or no error handling implemented.	**0—4** The values cannot be entered or the app crashes regardless of what the user enters.	
App Functionality: The calculations for the vehicle loan works correctly.	**16—20** The vehicle loan calculations work perfectly.	**10—15** The vehicle loan calculations most work correct.	**5—9** The vehicle loan calculations are implemented but mostly don't work correctly.	**0—4** The vehicle loan calculations are not implemented or don't work at all.	

RUBRIC 2	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of Task 2, students need to have:	**Excellent**	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks possible)				
App Functionality: The expenses are displayed in descending order according to value.	**9—10** The expenses are excellently displayed in descending order.	**7—8** The expenses are displayed in descending order, but the display could look better.	**4—6** The expenses are displayed but not in descending order according to value.	**0—3** The expenses are not displayed at all.	
Application Structure: The expenses are stored in a generic collection.	**9—10** The expenses are correctly stored in a generic collection.	**7—8** Most of the expenses are stored in a generic collection.	**4—6** Some of the expenses are stored in a generic collection.	**0—3** The expenses are stored in an array or a non-generic collection.	
Application Structure: The 75% expenses notification is implemented using a delegate.	**9—10** The 75% notification is excellently implemented using a delegate.	**7—8** The 75% notification is adequately implemented using a delegate.	**4—6** The 75% notification is adequately implemented something other than a delegate.	**0—3** The 75% expenses notification is not implemented at all or doesn't work at runtime	

RUBRIC 2	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of Task 2, students need to have:	**Excellent**	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks possible)				
Coding Standards: Code is well structured and documented.	**9—10** Code is well structured with good comments explaining logic.	**7—8** Code is well structured with minor mistakes and mostly commented.	**4—6** Code is structured somewhat well with little or no comments.	**0—3** Code is all in one file with no comments.	
Documentation: Readme file provides enough information to run the app.	**9—10** An excellent readme file is included that explains all the required details about running the app.	**7—8** The readme file presents some information about running the app but could be more detailed.	**4—6** The readme file contains information about running the app, but it is hard to understand or doesn't work.	**0—3** No readme file is included, or the readme file doesn't provide any useful information about running the application.	
Other Marks: Advanced features not covered in class (Bonus Marks).	**[5]** Excellent use of features that go above and beyond what was covered in class.	**[3—4]** Good use of features that go above and beyond what was learned in class was used.	**[1—2]** Some features that go above and beyond what was learned in class was used.	**[0]** No features that go above and beyond what was learned in class was used.	
TASK 2 SUBTOTAL					**/100**

RUBRIC 3 (POE)	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of the final Portfolio of Evidence, students need to have:	**Excellent**	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks possible)				
Updates: All the feedback provided on task 2 has been implemented.	**9—10** Excellent implementation of all feedback provided.	**7—8** Most feedback was implemented.	**4—6** Some feedback was implemented.	**0—3** Little or no feedback was implemented.	
App Functionality: User can enter values for income, tax and expenditures.	**9—10** All values can be entered, and good error handling implemented.	**7—8** All the values can be entered, but error handling could be improved.	**4—6** Only some of the values can be entered or no error handling implemented.	**0—3** The values cannot be entered or the app crashes regardless of what the user enters.	
App Functionality: Rental and home loan functionality working correctly.	**12—15** Rental and home loan functionality excellently implemented in the new user interface.	**8—11** Rental and home loan functionality implemented with some errors.	**4—7** Rental and home loan functionality implemented with lots of errors.	**0—3** Rental and home loan functionality not implemented or not working at all.	

RUBRIC 3 (POE)	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of the final Portfolio of Evidence, students need to have:	**Excellent**	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks possible)				
App Functionality: Car purchase working correctly.	**12—15** Car purchase functionality excellently implemented in the new user interface.	**8—11** Car purchase functionality implemented with some errors.	**4—7** Car purchase n functionality implemented with lots of errors.	**0—3** Car purchase functionality not implemented or not working at all.	
App Functionality: New savings or graph feature working correctly.	**15—20** New feature excellently implemented.	**10—14** New feature implemented with some minor errors.	**5—9** New feature implemented but causing lots of errors.	**0—4** New feature not implemented or not working.	
Usability: User interface is easy to use.	**9—10** User interface is excellently implemented and very easy to use.	**7—8** User interface is well implemented with a few small usability problems.	**4—6** The user interface can be used but is not very logical.	**0—3** User interface is completely confused and illogical.	

RUBRIC 3 (POE)	Levels of Achievement				Feedback
In order to be awarded full marks for these elements of the final Portfolio of Evidence, students need to have:	**Excellent**	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks possible)				
Documentation: User manual is well-structured with useful screenshots.	**12—15** Very complete user manual included.	**8—11** Mostly complete user manual included.	**4—7** Some information included.	**0—3** Not included or almost no detail.	
Other Marks: Readme file provides enough information to run the app.	**5** An excellent readme file is included that explains all the required details about running the app.	**3—4** The readme file presents some information about running the app but could be more detailed.	**1—2** The readme file contains information about running the app, but it is hard to understand or doesn't work.	**0** No readme file is included, or the readme file doesn't provide any useful information about running the application.	
Other Marks: Student has used advanced features not covered in class (Bonus Marks).	**5** Excellent use of features that go above and beyond what was covered in class.	**3—4** Good use of features that go above and beyond what was learned in class was used.	**1—2** Some features that go above and beyond what was learned in class was used.	**0** No features that go above and beyond what was learned in class was used.	
POE SUBTOTAL					**/100**