MODULE NAME:	MODULE CODE:
PROGRAMMING 1B	PROG6112

ASSESSMENT TYPE: ASSIGNMENT 1 (PAPER ONLY)

TOTAL MARK ALLOCATION: 100 MARKS

TOTAL HOURS: 15 HOURS

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. ***Make a copy of your assignment before handing it in.***
3. *Assignments must be typed unless otherwise specified.*
4. *All work must be adequately and correctly referenced.*
5. *Begin each section on a new page.*
6. *Follow all instructions on the assignment cover sheet.*
7. *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:

....

....

Question 1 (Marks: 65)

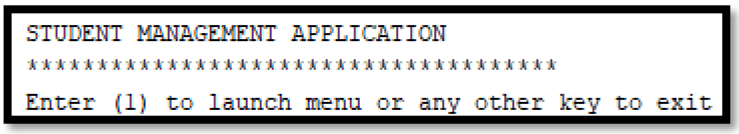
Section A (Marks: 40)

ABC College is a local education provider that specialises in software development, information management, and mobile application development training. The educational provider has recently opened a college in your town and has hired the software development house you work for to design a Java application to manage their students.

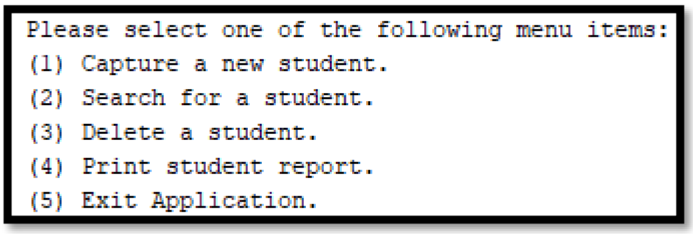
Your line manager has requested you develop the application with the following requirements:

1.1. When the application starts, it must display the following menu structure:

Sample Menu Screen Shot



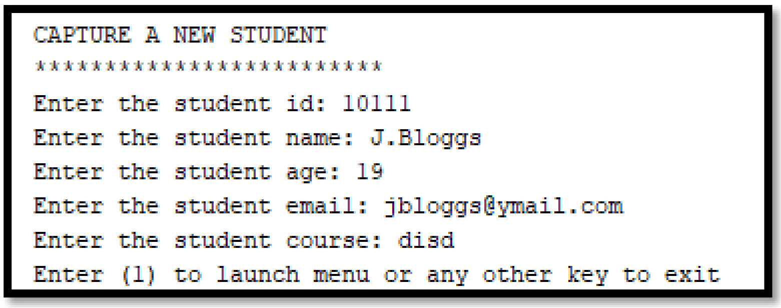
```
STUDENT MANAGEMENT APPLICATION
****************************************
Enter (1) to launch menu or any other key to exit
```



```
Please select one of the following menu items:
(1) Capture a new student.
(2) Search for a student.
(3) Delete a student.
(4) Print student report.
(5) Exit Application.
```

1.2. If the user selects to capture a new student, you must save all the information supplied by the user to memory. You may use arrays or array lists to save the student model to achieve this.

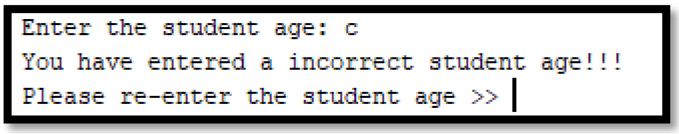
Sample Capture Student Screen Shot



```
CAPTURE A NEW STUDENT
*************************
Enter the student id: 10111
Enter the student name: J.Bloggs
Enter the student age: 19
Enter the student email: jbloggs@ymail.com
Enter the student course: disd
Enter (1) to launch menu or any other key to exit
```

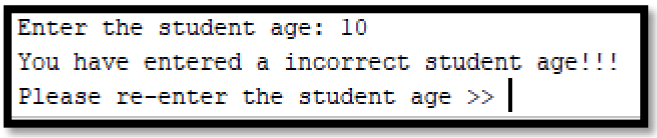
1.3. If the user enters an incorrect student age, prompt the user to re-enter a valid one. A valid student age is any age greater than or equal to 16. Only numbers are allowed to be supplied when entering a student's age.

Sample Screen Shot of an invalid student age character.



```
Enter the student age: c
You have entered a incorrect student age!!!
Please re-enter the student age >>
```

Sample Screen Shot of an invalid age.

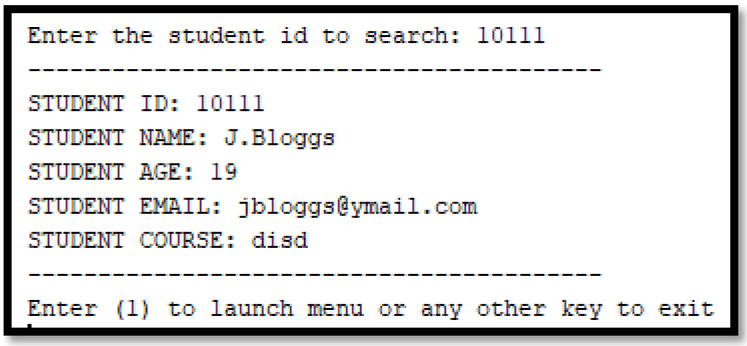


```
Enter the student age: 10
You have entered a incorrect student age!!!
Please re-enter the student age >>
```

1.4. Once the entire submission has been completed, the user must be informed that the student details have been successfully saved.

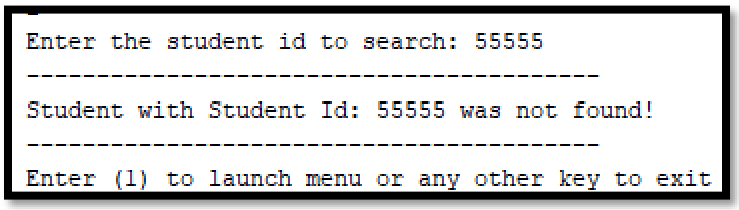
1.5. The user must have the ability to search for a student. The user will select menu item two, which will prompt the user to enter a student ID. If a student is found in the application, display the student's details to the user. If no student is found, display an error message to the user that the student cannot be located.

Sample Student Search Screen Shot



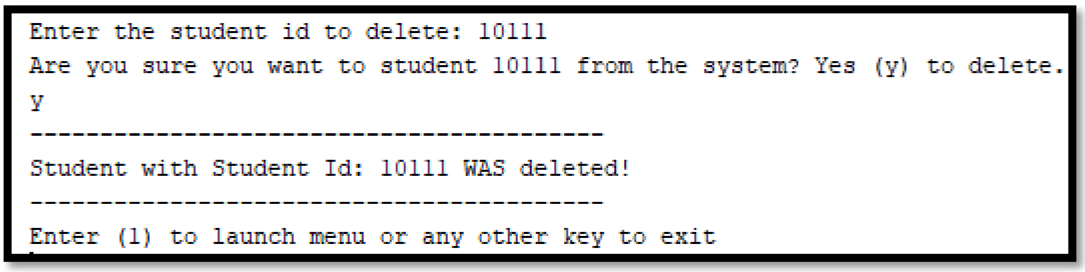
```
Enter the student id to search: 10111
-----------------------------------------
STUDENT ID: 10111
STUDENT NAME: J.Bloggs
STUDENT AGE: 19
STUDENT EMAIL: jbloggs@ymail.com
STUDENT COURSE: disd
-----------------------------------------
Enter (1) to launch menu or any other key to exit
```

Sample Invalid Student Screen Shot



```
Enter the student id to search: 55555
-----------------------------------------
Student with Student Id: 55555 was not found!
-----------------------------------------
Enter (1) to launch menu or any other key to exit
```

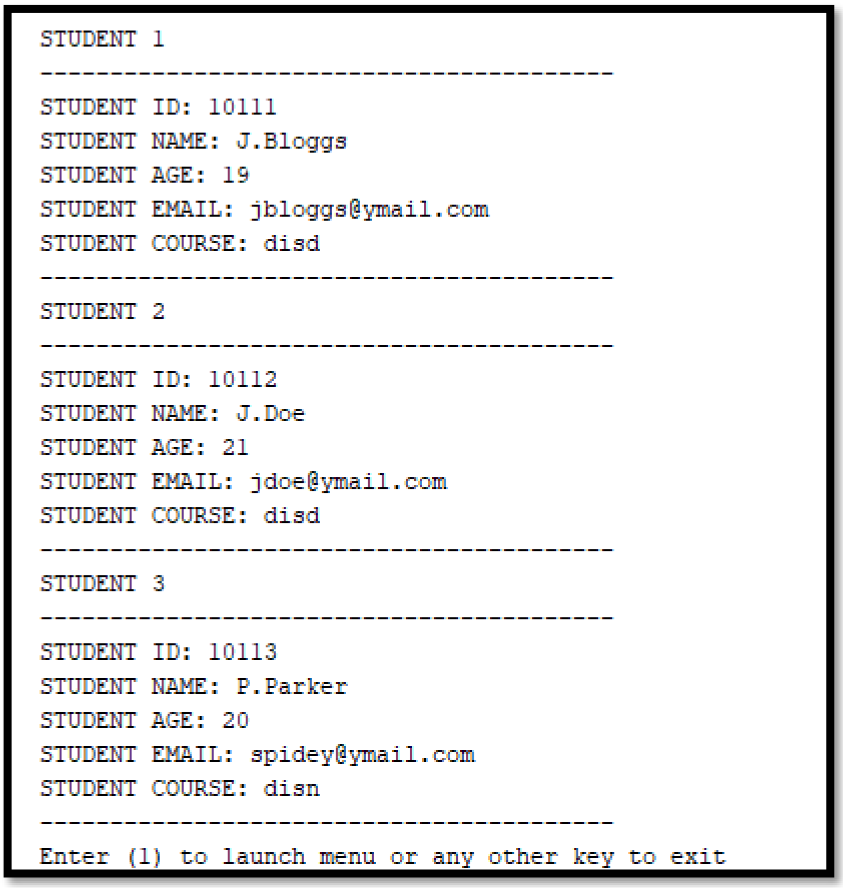
1.6. The user must have the option to delete a student that has been saved. The user must first enter the student ID to be deleted. The user must confirm whether they want to delete the student.



```
Enter the student id to delete: 10111
Are you sure you want to student 10111 from the system? Yes (y) to delete.
y
-----------------------------------------
Student with Student Id: 10111 WAS deleted!
-----------------------------------------
Enter (1) to launch menu or any other key to exit
```

1.7. When the user selects to view the student report, display the report below, which is generated from the memory collection in your application.

Sample Report Screen Shot



```
STUDENT 1
-----------------------------------------
STUDENT ID: 10111
STUDENT NAME: J.Bloggs
STUDENT AGE: 19
STUDENT EMAIL: jbloggs@ymail.com
STUDENT COURSE: disd
-----------------------------------------
STUDENT 2
-----------------------------------------
STUDENT ID: 10112
STUDENT NAME: J.Doe
STUDENT AGE: 21
STUDENT EMAIL: jdoe@ymail.com
STUDENT COURSE: disd
-----------------------------------------
STUDENT 3
-----------------------------------------
STUDENT ID: 10113
STUDENT NAME: P.Parker
STUDENT AGE: 20
STUDENT EMAIL: spidey@ymail.com
STUDENT COURSE: disn
-----------------------------------------
Enter (1) to launch menu or any other key to exit
```

1.8. Finally, provide the ability for the user to exit the application.

Additional Requirements:

1.9. In your solution, you are required to create a class called Student which will contain all your working methods.

1.10. This class will, as a minimum, contain the following methods, but you are encouraged to add more methods:

- SaveStudent();

- SearchStudent();
- DeleteStudent();
- StudentReport();
- ExitStudentApplication ();

1.11. Create a main class to run your application.

Section B (Marks: 25)

Your line manager has instructed you to write unit tests to determine the code coverage in the student management application. You are required to create a test package within the application you created in **Section A,** which will contain the required unit tests.

You are required to write the following unit tests:

Test Name	**Test Purpose**
TestSaveStudent()	To supply the student ID, name, age, email, and course to the save student method. The test will determine whether the correct student will be saved to memory.
TestSearchStudent ()	To supply the student id to the search student method. The test will determine that the correct student details have been returned.
TestSearchStudent_StudentNotFound()	To supply an incorrect student ID to the search student method. The test will determine that no student was found.
TestDeleteStudent()	To supply the student ID to the delete student method. The test will determine that the student has been successfully deleted.
TestDeleteStudent_StudentNotFound()	To supply an incorrect student ID to the delete student method. The test will determine that no student could be found to delete.

TestStudentAge_StudentAgeValid()	To supply a valid student age to the student age method. The test will determine that the student's age is valid.
TestStudentAge_StudentAgeInvalid()	To supply an invalid student age to the student age method. The test will determine that the student's age is less than 16.
TestStudentAge_StudentAgeInvalidCharacter()	To supply an invalid character to the student age method. The test will determine that the student age supplied is not a number.

Question 2 **(Marks: 35)**

You have to design your own Java console application for any valid problem that your application must solve. Your solution can include solving a business problem, a new idea, or a game. Your application must use concepts such as arrays, loops, inheritance, constructors, and information hiding. Output must be shown in the form of a report using the console. You must also write unit tests to validate that your code achieves the desired result.

In your solution, make use of as many concepts and components dealt with in this course, but pay special attention to the following learning units:

- Learning Unit 1: Advanced arrays
- Learning Unit 2: Introduction to Inheritance

END OF PAPER

Appendix A

Assessment Sheet (Marking Rubric)

MODULE NAME:	MODULE CODE:
PARENT MODULE NAME	PROGRAMMING 1B
CHILD MODULE NAME	PROG6112

STUDENT NAME:	
STUDENT NUMBER:	

Question 1: Section A - Mark Allocation	Levels of Achievement				Feedback
	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ marks possible)				
Variables declared and assigned.	5	2-4	1	0	
Menu and layout created.	5	2-4	1	0	
Saving the captured values to memory	5	2-4	1	0	
Searching for a student	5	2-4	1	0	
Deleting a student	5	4	2-3	0	

Updating a student	**5**	**4**	**2-3**	**0**	
Student report generated	**3**	**2**	**2**	**0**	
Student class created with working methods	**3**	**2**	**1**	**0**	
Good programming practice and comments	**4**	**2-3**	**1**	**0**	

Question 1: Section B - Mark Allocation	**Levels of Achievement**				**Feedback**
	Excellent	**Good**	**Developing**	**Poor**	
	Score Ranges Per Level (½ marks possible)				
TestSaveStudent unit test written with the correct test result	**3**	**2**	**0-1**	**0**	
TestSearchStudent unit test written with the correct test result	**4**	**3**	**1-2**	**0**	
TestSearchStudent_StudentNotFound unit test written with the correct test result	**3**	**2**	**0-1**	**0**	
TestDeleteStudent unit test written with correct test result	**3**	**2**	**0-1**	**0**	
TestDeleteStudent_StudentNotFound unit test written with the correct test result	**3**	**2**	**0-1**	**0**	
TestStudentAge_StudentAgeValid unit test written with the correct test result	**3**	**2**	**0-1**	**0**	
TestStudentAge_StudentAgeInvalid unit test written with the correct test result	**3**	**2**	**0-1**	**0**	
TestStudentAge_StudentAgeInvalidCharacter unit test written with the correct test result	**3**	**2**	**0-1**	**0**	

Question 2 Mark Allocation	Levels of Achievement				Feedback
	Excellent	Good	Developing	Poor	
	Score Ranges Per Level (½ marks possible)				
Variables declared.	**3**	**2**	**0-1**	**0**	
Console application created with acceptable layout.	**4**	**2-3**	**1**	**0**	
Concepts such as arrays, loops, inheritance, constructors, and information hiding.	**4**	**2-3**	**1**	**0**	
Code for application	**5**	**3-4**	**2-3**	**0-1**	
Creative application with more than basic functionality and effort	**4**	**2-3**	**1-2**	**0-1**	
Unit tests created to prove code functions as intended	**15**	**10-14**	**5-9**	**0-4**	