ASSESSMENT NOTIFICATION

Subject	Software Design and Development
Course Teachers	Liam Dunphy
Learning Leader	Liam Dunphy
Date of distribution of this notification	Term 4 Week 1

TASK DETAILS

Name of Assessment Task	Minor Project
Task Weighting (%)	15%
When is the task due?	Term 4, Week 7 Saturday 20th November, submitted online by 9pm
How do I submit the task?	• Submit digital copy via CANVAS.NBCS • Online journal for weekly posts
Word Limit	*INSERT WORD LIMIT*
Outcomes to be assessed	H1.1, H2.2, H5.1, H5.2
Type of task	• Written document containing Software Design diagrams • Online Journal for weekly progress posts
How do I complete this task?	• Submit the final document in .pdf format • Submit weekly progress updates in the online journal • Submit the final program (code and application)
Where can I get help?	• Mr Dunphy • Canvas ○ Inbox ○ Discussions – online journal • Online forums, e.g. StackOverflow, GitHub, CodePen, w3schools

ASSESSMENT EXPECTATIONS

- Assessment tasks must be submitted on time. Late submissions will be penalised by up to 25% of the available marks per day. Extensions are coordinated by the Learning Leader.
- Assessment tasks must be entirely your own work. Plagiarism or other forms of academic malpractice will most likely result in a mark of zero and the requirement to resubmit the task.
- Referencing is to utilise the Harvard referencing method. Instructions for this referencing style are found in the Assessment Guide for each grade.

TASK DESCRIPTION

Overview

This task is designed to test and extend your knowledge and skills in designing and developing software projects in *a programming language of your choice*. It is anticipated that in the course of the task you will need to continue to further your knowledge to complete the various projects. You will need to draw on all the programming concepts taught thus far. When designing your interface please keep in mind all the aspects of:

1. ease-of-use
2. consistency
3. ergonomics

Deliverables:

For your project, you must provide the following items:

A) Project Documentation

- A **title page** displaying a project name, your name and image
- A short (150 word) **description** of:
  - the program
  - **end users**
  - program **requirements**.
  - The following System Modelling Tools:
    - **System Flow Chart**
    - **Data Flow Diagram**
    - **IPO diagram**
    - **Data Dictionary**
    - **Structure diagram** for your program.

      N.B. These should be completed as part of your weekly course work throughout this term.
  - **Story Boards** and **Screen Designs** for your game.
  - A **Flowchart or Pseudo code** of "*Can't Lose*" algorithm for your program.

B) Source code

- from your main game and any other **forms** or **classes** you create.

C) Weekly Journal posts - Planning and Progress reporting

- At least **5 weekly updates** about your progress on the online Assessment 1 Journal. (Must be at least 2 Days Apart and posted in separate weeks of Terms as a progress/planning report)

D) A working application.

e.g.

- ***.html*** , ***.cs*** , ***.js*** for JavaScript
- ***.exe*** *for C#*
- ***.py*** *for python***)**

What you actually hand in:

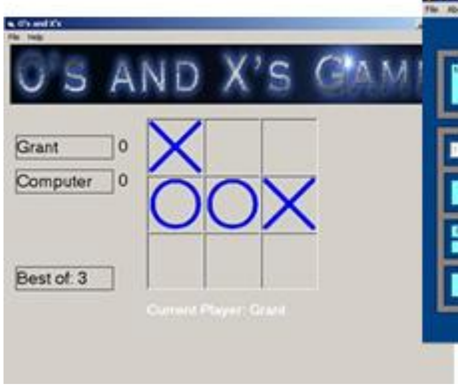
1. Items A and B are to be submitted in a single file*.
2. Item C should be linked in your Project Documentation (Item A) but will be marked online in the Canvas Discussion for your online journal.
3. The application at item D is to be directly uploaded into the online CANVAS learning environment in its appropriate form, e.g. zipped folder, if using JavaScript

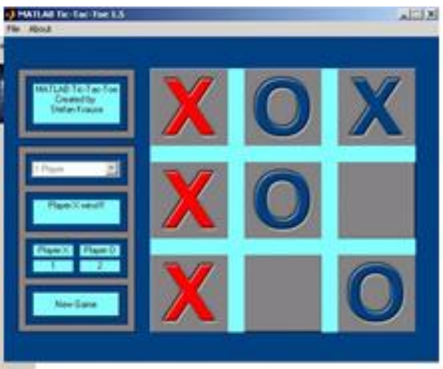
   *the uploaded documentation file may be either a Microsoft Word (.doc) file, an OpenDocument Text (.odt) file, a Portable Document Format (.pdf) file or a Google Doc date stamped prior to the submission date.

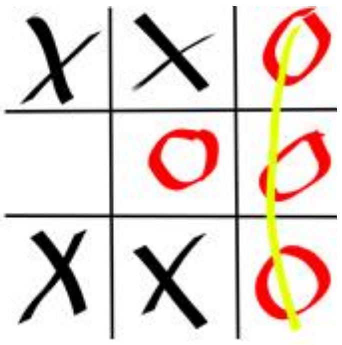
Project Details

Tic-Tac-Toe Game (25 marks – see later for the breakdown)

Tic-tac-toe, also called **noughts and crosses**, **hugs and kisses**, and many other names, is a game for two players, **O** and **X**, who take turns marking the spaces in a 3×3 grid, usually **X** going first. The player who succeeds in placing three respective marks in a horizontal, vertical or diagonal row wins the game. Visit **Wikipedia** for more information on this game if you do not already know it!







Required Application Elements:

Design and produce a software package that enables 2 people to play "Naughts and Crosses" or "Tic-Tac-Toe" (different names for the same game) against each other. View the online example (unfinished) as a guide. Your game should include the following elements:

- The standard application menus, ergonomic expectations, and attention to detail.
- **Element 1**: A simple single-form game which two people can play. The computer detects and displays the winner. **(40% *of total*)**
- **Element 2**: A multi-form game which uses:
    a. a start-up screen to select player names and "Best Of" settings for the main game which is then as in Element 1.
    b. Player's names, scores and turn are displayed at game time **(50% *of total*)**
- **Element 3**: As with element 2 but including the functionality to *play a simple computer opponent* which randomly moves in its turn. **(70% *of total*)**
- **Element 4**: A complete game including elements 1- 3 above as well as a *"Easy/Hard" setting* on the start form when playing the computer opponent. The "Hard" setting should use the "**Can't Lose!**" algorithm given online to calculate where to "move" on the computers turn making the computer unbeatable. **(100% *of total*)**

Required Coding Elements

Various programming techniques learnt in the course so far must be implemented in your source code somewhere, these include:

1. The use of:
    a. Selection ("If", "Select... Case")
    b. Iteration ("For", "While" or "Do while")
    c. Array or List.
2. Extensive use of **Methods/Functions** to modularise your code thus improving maintainability.
3. The creation and implementation of at least **one class** (*if using a language such as C# or JavaScript)* as you see fit.

Your program should also display the following generally accepted good coding practices:

1. Extensive use of **comments** in your source code to describe what you are doing
2. Intrinsic documentation - Use of **meaningful and clear Variable** and Object/Function names.
3. Clear and **well formatted code**. (i.e. Code is not all in one straight line, you must use tabbing for clarity with indentation.)

Extended Elements (Extra 2 marks... if required)

The following is a list of elements which you can include into your program to receive bonus marks if you lose marks elsewhere in your project:

1. Ability to save a game mid-way through and then start it again later. (Using an external text file)
2. Any other feature that you feel would improve the "gaming experience". A written justification must be written in your project documentation to receive this bonus mark.
3. Integrated (with working links) into the Software Portfolio website that you made for your Y11 Major Project in a page titled "HSC Minor Project".

Documentation Details

Modelling – Representing a System Using Diagrams

- (25 marks – see later for the breakdown)

As you may notice the documentation for this project counts as much as the program itself and it is therefore vital that you complete all aspects required for this document. During the course you will be introduced to the notion of modelling a software system using a variety of different diagrams. These diagrams are presented on **pages 121-144** of your textbook and are also described in the NESA SDD Course Specifications document. They will also be explicitly addressed throughout this term. It is your job to construct one of each of these diagrams for your Tic-Tac-Toe system.

The following is a list of required elements and guidelines for the documentation:

1. The submitted document* must contain elements A through C described previously and adopt these guidelines:
    - Submitted in a **typed format using no larger than 12 point font and 1½ line spacing (**with acceptable margins, e.g. default margin settings).
    - **Title page** with your name, the name of the task with a relevant image and the date of the submission
    - **Items A through C as described previously in this document.**
    - (optional) **Appendices** – all reference materials used in the completion of the task (e.g. interview notes, etc)
      N.B. the submission will use TurnItIn to detect the source of any reference material used so it is expected therefore that you will reference anything that TurnItIn might find.
    - **Page numbers** on every page (except perhaps the title page)
2. **Flowchart Note**: Even if you have not been able to implement the "**Can't Lose**" algorithm into your program you are still required to draw a flow chart of the algorithm for your documentation as a modelling tool of your plan to implement the algorithm. (A description of this algorithm can be found online.)
3. The **5 required journal posts**** which you need to complete online are all under one discussion which should be started using:
    a. the title "*Students Name*'s Assessment 1 Journal".
    b. All subsequent journal updates for your project should be made as "Replies" to your original message.
    c. Your forum entry should highlight:
        - your current progress in the project (where you are up to)
        - current struggles in the project
        - any other reflections which you might have.
        - The post should also describe the plan for tasks to be completed for the week ahead.

MARKING RUBRIC

Documentation

- How clear, concise is the overview?
- Is the language appropriate to the context?
- Have you applied project management techniques to maximize the productivity of the software development?

(25 Marks)	Thorough Bands 5-6	Sound Bands 3-4	Basic Bands 1-2
Title page and document presentation	A title page is included with your name, title and image on it, which is separate from the rest of the document. The document is professional and contains screenshots, including page numbers and table of contents. **(1)**		
Description	A **clear description** of the ***needs, objectives, boundaries,*** and ***user's requirements*** is given. **(2)**	A short or unclear description of the program is given. **(1)**	Not provided **(0)**
System Flowchart	**System Flow Chart** is detailed and uses the correct symbols. **(3)**	System Flow Chart contains some relevant details or mostly uses the correct symbols. **(2)**	System Flow Chart is basic or uses incorrect symbols. **(1)**
Data Flow Diagram	**Data Flow Diagram** clearly/correctly describes the flow of data through the system. **(3)**	Data Flow Diagram describes the flow of data through the system. Some errors. **(2)**	An attempt has been made at the Data Flow Diagram. **(1)**
IPO Diagram	**IPO diagram** produced is correct. **(3)**	IPO diagram produced with some errors. **(2)**	An attempt has been made to produce an IPO diagram **(1)**
Structure Chart	**Structured Diagram** identifies the sequence and levels of processes in the system correctly. **(3)**	Structured Chart attempted with some errors. Mostly correct. **(2)**	An attempt has been made at the Structured Chart. **(1)**
Data Dictionary	**Data Dictionary** is present and correct. **(3)**	Data Dictionary is produced with some errors. **(2)**	An attempt has been made to produce a Data Dictionary. **(1)**
Program Flowchart	**Program Flowchart** of "Can't Lose" algorithm uses correct symbols and is logically correct. **(4)**	Program Flowchart of "Can't Lose" algorithm uses correct symbols with some logical errors. **(2-3)**	Limited attempt at producing a program flowchart. **(1)**
Storyboard and Screen Design	**Storyboard** clearly shows the interfaces (**screens**) in the game and the links between them. Elements of each Screen are clearly identified and the links between screens are clearly shown. **(3)**	**Storyboard or Screen design** attempted with some errors. Mostly correct. **(2)**	An attempt has been made at the **Storyboard or Screen design**. **(1)**
Journal Posts and Gantt Chart	5 or more **journal entries** at least 2 days apart record/describe the progress of the program. The Journal posts are in sync with the schedule in the Gantt Chart **(5)**	3- 4 journal discussions are included describing the progress of the project. There may be discrepancies between the Gantt Chart and the forum posts **(3-4)**	1- 2 forum discussions are included describing the progress of the project. Limited attempt to schedule the project tasks. **(1-2)**

Code Design

- Clear & appropriate variable names?
- Use of comments to internally document complex elements.
- Use of code structure techniques.

(15 Marks)	Thorough Bands 5-6	Sound Bands 3-4	Basic Bands 1-2
Maintainability of code	Naming in compliance with the good programming practice conventions. Extensive use of comments balanced by appropriate intrinsic documentation. **(3)**	Somewhat logical naming evident. Some use of comments. **(2)**	Weak naming (eg. many variables and controls still have their original names (e.g. button1, textBox1). Minimal use of comments. **(1)**
Coding Structure	All three required coding structures **(Selection, Iteration and Arrays)** are present and use correctly. **(3)**	Two of the three required coding structures **(Selection, Iteration and Arrays)** are present and use correctly. **(2)**	Only one coding structure **(Selection, Iteration or Arrays)** has been used correctly. **(1)**
Methods/Functions *(Abstraction)*	Extensive and logical use of **Methods/Functions** to break problem into smaller problems. **(3)**	Substantial use of **Methods/Functions** to break problem into smaller problems. **(2)**	Minimal use of **Methods/Functions** to break problem into smaller problems. **(1)**
Classes/Objects (OOP data structure)	A **Class** is constructed and **Objects** of that class are logically used within the program. **(3)**	A **Class** is constructed and **Object(s)** of that class are used within the program. (Some errors). **(2)**	An attempt is made at constructing classes and objects. **(1)**
Efficient use of code	The code displays a good understanding of efficient programming practice **(3)**	The code displays some understanding of efficient programming practice **(1)**	The code displays a limited understanding of efficient programming practice **(1)**

Working Application

- Does the application perform as required?
- Is the user input validated?

• **(10 Marks)**	• Thorough • Bands 5-6	• Sound • Bands 3-4	• Basic • Bands 1-2
Element 1	Application **Element 1** is completed successfully and working. **(3)**	Some aspects of element 1 function appropriately without error. **(2)**	An attempt has been made at element 1. **(1)**
Element 2	Application **Element 2** is completed and working. **(2)**	Some attempt at element 2 is evident but not working. **(1)**	No attempt made to create element 2 **(0)**
Element 3	Application **Element 2** is completed and working. **(2)**	Some attempt at element 3 is evident but it does not work properly. **(1)**	No attempt made to create element 3 **(0)**
Element 4	Application **Element 4** is completed. The application is error free and has a high attention to detail. **(3)**	Some attempt at element 3 is evident but not working. **(2)**	An attempt at element 4 has been made but is not working correctly. **(1)**
Extended Elements	A file (e.g. .txt) is used to save all game data. This can be easily used to reload a game from within the program. --- AND/OR --- A notable addition to the game has been made and is working. --- AND/OR --- Integrated into the Software Portfolio (Y11 Major Project website) with working links in the submitted zipped folder **(2)**	An attempt has been made to save the game but there are some errors. --- OR --- A notable addition to the game has been made and is working. ---- OR --- Integrated into the Software Portfolio (Y11 Major Project website) with working links in the submitted zipped folder **(1)**	Not Attempted **(0)**

HSC Outcomes

H1.1	explains the interrelationship between hardware and software
H1.2	differentiates between various methods used to construct software solutions
H1.3	describes how the major components of a computer system store and manipulate data
H2.1	explains the implications of the development of different languages
H2.2	explains the interrelationship between emerging technologies and software development
H3.1	identifies and evaluates legal, social and ethical issues in a number of contexts
H3.2	constructs software solutions that address legal, social and ethical issues
H4.1	identifies needs to which software solutions are appropriate
H4.2	applies appropriate development methods to solve software problems
H4.3	applies a modular approach to implement well structured software solutions and evaluates their effectiveness
H5.1	applies project management techniques to maximise the productivity of the software development
H5.2	creates and justifies the need for the various types of documentation required for a software solution
H5.3	selects and applies appropriate software to facilitate the design and development of software solutions
H6.1	assesses the skills required in the software development cycle
H6.2	communicates the processes involved in a software solution to an inexperienced user
H6.3	uses and describes a collaborative approach during the software development cycle
H6.4	develops and evaluates effective user interfaces, in consultation with appropriate people

PERFORMANCE BAND DESCRIPTIONS FOR SOFTWARE DESIGN AND DEVELOPMENT

The typical performance in this band:

Band 6

- demonstrates a thorough understanding of the phases of the software development cycle in producing a solution relevant to client needs and concerns
- uses appropriate development methodologies and project management techniques to analyse a problem and design a complete software solution
- develops well-constructed algorithms for a variety of unfamiliar problems using appropriate control structures and data structures
- effectively uses appropriate resources, tools and documentation to manage the development and to communicate the essential features of software solutions
- designs an effective software solution to a problem reflecting a sophisticated understanding of the interrelationships between hardware and software
- critically evaluates the social and ethical issues related to the development of software solutions and the impact on society of the use of computer-based solutions
- analyses the effects of historical developments on current and emerging technologies and practices, and the development process

Band 5

- demonstrates an understanding of the phases of the software development cycle in producing a solution recognising client needs and concerns
- uses development methodologies and project management techniques to analyse a problem and design a relevant software solution
- develops algorithms for a variety of problems using appropriate control structures and data structures
- uses a variety of resources, tools and documentation to manage the development and to communicate the essential features of software solutions
- designs a software solution to a problem reflecting an understanding of the interrelationships between hardware and software
- makes informed judgement about the social and ethical issues related to the development of software solutions and the impact on society of the use of computer-based solutions
- relates knowledge and understanding of historical developments to current and emerging technologies and practices, and the development process

Band 4

- outlines the phases of the software development cycle required to produce a solution to a specified problem
- uses development methodologies and project management techniques to design a software solution
- develops an algorithm for a specified problem showing some understanding of control structures and data structures
- uses a limited number of resources, tools and documentation to develop and communicate some features of software solutions
- modifies a software solution to a problem reflecting knowledge of the interrelationships between hardware and software
- gives a clear explanation of the impact on society of the use of computer-based solutions
- demonstrates knowledge and understanding of historical developments and current and emerging technologies

Band 3

- demonstrates a basic understanding of the phases of the software development cycle
- designs a partial software solution recognising the need for project management techniques
- reads, interprets and modifies simple algorithms that use a variety of data structures
- recognises and describes some resources, tools and documentation used to develop and communicate software solutions
- demonstrates a basic knowledge of the interrelationships between hardware and software
- describes some issues related to the impact on society of the use of computer-based solutions
- shows some knowledge

Band 2

- identifies the phases of the software development cycle
- recognises some development methodologies and project management techniques
- reads and interprets simple algorithms that use simple data types
- recognises some resources, tools and documentation used in software development
- distinguishes between hardware and software components
- identifies some issues related to the impact on society of computer-based solutions
- identifies examples of current technologies

Band 1