CSY1063: Web Development					
Due for Issue (week commencing):	**05/02/2024**		**Last Date for Submission:**		**19/05/2024 (23:59)**
Aspect (& weighting)	Excellent A	Good B	Satisfactory C	Needs some more work D	Needs much more work F
Functionality (60%)					
Evaluation (10%)					
Testing (10%)					
Code Quality and Efficiency (15%)					
Video Demonstration (5%)					

Submission Guidelines – Please read carefully

1. The University of Northampton's Policy on Plagiarism & Mitigating Circumstances will be strictly implemented.
2. This is not a group project, by submitting this assignment you are asserting that this submission is entirely your own individual work. You may discuss the assignment with other students but any code written should be your own. **Sharing your work with another student, or submitting code that was written by someone else may be deemed academic misconduct.**
3. If you have used any code that you did not write you must: i. Correctly reference the code in the report page (use Harvard referencing) ii. In your report page, clearly document which lines of code you used, where you used them and what they are used for.
4. You must supply **all** items of assessment and **upload them to the correct submission points**.
    I. Source code in a zip file. The marker must be able to download and run your code. Include the report with your zip file which contains a URL to your video demonstration.
    II. Source code word document (uploaded to turnitin)

Please make sure you double check all submissions. It is your responsibility as a student to ensure these guidelines have been met.

Failure to follow the submission guidelines may result in a capped grade of F-.

If the marker has any questions you may be asked to attend an online viva to discuss your submission.

Assignment 2 (JavaScript)

Aims & Objective

The purpose of this assignment is to assess your ability to create an interactive web page using JavaScript, CSS and HTML.

Brief

The zip file as2.zip provides the start of a game with existing functionality for:

- Moving the player around using the arrow keys

The aim of the game is for the player to collect all the points within the maze. Once all the points are collected, the game will end. The player should not be able to move through the walls of the maze. Additionally, the game should end if the player encounters the enemy (represented by green circle elements).

Two CSS classes are provided for the player:

- hit (display a hit animation for the player)
- dead (display a death animation for the player)

If the player does touch an enemy, the player should lose a life (top right of the screen). When the player has lost all the lives the game should end with an option to restart.

Requirements

For a **bare pass** (D- - D+) you must implement the following:

1) When the start button is pressed, the game should begin, and the start button should no longer be visible.
2) The player should not be able to move until the start button has been pressed.
3) Stop the player from walking through the maze's walls (blue squares).
4) When the player touches a point (small white circle element), the point is removed from the maze.
5) Update the score for every point the player collects (<p> tag in .score).
6) A game-over message will appear when the player collects all the points.
7) If the player touches an enemy (green circle element), a game-over message will appear, and the player will display the dead animation (css class .dead).

For a **good pass** (C- - B-) you must improve the game

1) Randomise the position of enemies at the start of the game.
2) Prevent the enemies from being created outside the maze and only allow them to appear where there is a free space (0 in the maze array).
3) The enemies randomly move around in the maze. If they hit a wall, they should move in a new direction (they should not get stuck).
4) Enemies stop moving when the game-over state has been reached.
5) Instead of the game-over message display "Restart?" to allow the player to restart the game if they lose, the game should reset back to its original state upon clicking restart.
6) Implement the arrow buttons. The player will continue moving in that direction when an arrow button is clicked. Only one arrow direction can be used at a time. The player should only move in that direction.

For a **very good pass** (B - A-)

1) At the end of the game, ask the player to enter their name (use a prompt()) and save their name along with their score using local storage.
2) Display the five highest player scores in the leaderboard (.leaderboard div).
3) The leaderboard should be organised in order from the highest score to the lowest score.
4) Add the player's lives using JavaScript (not HTML) at the start of the game.
5) When the player collides with an enemy, remove a life instead of ending the game. The player should use the hit animation (.hit css class) and be unable to move for 1.5 seconds while the animation is being played.
6) If all three lives are lost, display the game-over/restart message.

For an **excellent pass** (A – A+) you must implement the following

1) Add levels of increasing difficulty. Once the player has collected all the points in the game, a new maze layout is shown. The difficulty can be increased by adding new enemies for each level. For example, level 1 has one enemy, while level 5 has five. As the levels increase, so too should the difficulty.
2) Bonus marks are available if there are an infinite number of levels. These are not premade mazes; the maze layout is automatically randomised each time.
3) Add two unique features to the game—the more complex the features, the higher the mark. For example, a power-up or a character customizer could be implemented. State what the additional features are in your report.

Technical Details

The following CSS classes are available for you to use.

Base	Secondary Actions
#player	.hit .dead
.point	
.enemy	

Deliverables

Along with your code, you will need to provide a report which includes the following sections:

1. **Checklist of completed functionality**
    a. A checklist of features implemented from the requirements section
2. **Testing**
    a. How did you test that your code worked?
    b. Could you test certain aspects of the code without running the entire game and waiting for the correct condition to be met?
    c. What tests did you carry out and what were the outcomes?
    d. What bugs did you discover during testing?
3. **Evaluation**
    a. A list of known bugs/weaknesses in the game
    b. What works well?
    c. What improvements could can be made?
    d. What else would you have done if you had more time?
    e. How easy would it be to extend the game to add more functionality?
    f. If you had to build a similar game in the future, what would you do differently and why?

4. **Video Demonstration URL**
    a. A link (URL) to your video demonstration

In Addition to the report you must provide a video demonstration of your assignment working. The demo should be 2-5 minutes (No longer than 10 minutes). The video should demonstrate features you have implemented.

Submission procedure checklist

1. All HTML, CSS, Images, JavaScript and the report in a zip file uploaded on NILE. **The marker must be able to download, extract and run your code, or your submission will be classed as incomplete**. You must include your technical report as a Word document in this zip file. There must be a URL to your video demonstration in your technical report.
2. All HTML, CSS and JavaScript code copied to a Word document (You must paste the code, not screenshots, from your editor. It doesn't matter if you lose syntax colouring)
3. Your JavaScript code must be your own work. The University of Northampton Policy on Plagiarism & Mitigating Circumstances will be strictly implemented. **By submitting this assignment, you assert that this submission is entirely your own/individual work**.
4. Excluding the supplied HTML/CSS and lecture notes, any code you submit that you did not write must be referenced. In your report, you must make it clear which sections of the code you didn't write and reference them correctly.

Failure to comply with the submission procedure, e.g. not uploading a source code appendix or video may result in a capped grade of an F-.

You must use the assets provided in the as2.zip, not using the assets will result in a capped grade of an F-.

Marking Criteria

Aspect (& weighting)	Excellent A	Good B	Satisfactory C	Needs some more work D	Needs much more work F
Functionality (60%)	Contains all functionality required for a B grade along with the functionality listed under the "For a very good pass" section of the assessment brief	Contains all functionality required for a C grade along with all the functionality listed under the "For a good pass" section of the assessment brief	Contains all functionality required for a D grade along with some functionality listed under the "For a good pass" section of the assessment brief	Contains all functionality listed under the "For a bare pass" section of the assessment brief	Does not meet requirements for a bare pass
Evaluation (10%)	All questions in the assessment brief have been answered with a good level of detail. Possible structural enhancements have been noted.	All questions in the assessment brief have been answered with a good level of detail. Some thought has been put into different ways the code could be structured.	All questions in the assessment brief have been answered with a satisfactory level of detail	Does not answer all questions asked in brief	Nothing of value has been presented to document the thought process behind the structure of the code
Testing (10%)	Everything for B but also has a strategy that makes testing easier. E.g. being able to speed up the race so you don't have to wait for the entire race to complete each time.	Tests have been provided for most of the functionality. Any bugs found are listed and fixed or an evaluation of what needs to be done to fix the bug	Tests have been provided for most of the functionality. Any bugs found are listed.	Some tests have been documented for some of the functionality	Little to no testing strategy has been provided
Code Quality and Efficiency (15%)	Use of functions, loops, arrays and other language tools to prevent repeated code.. Adding more enemies would require very minimal changes to the java script file	Everything for C but some attempt has been made at reducing repeated code.	Program meets basic requirements, minimal use of loops, functions to reduce repeated code.	Program meets basic requirements but code is repeated throughout and/or some bugs remain	Program does not work or contains errors which prevent the basic requirements being met.
Video Demonstration (5%)	Demonstrates multiple outcomes e.g. winning/losing	Demonstrates known bugs and several different iterations of the game being played.	Shows basic functionality but does not demonstrate known bugs	Shows basic functionality but does not demonstrate known bugs	Does not show the website meeting the basic functionality

Note: An A+ requires exceptional work which goes above and beyond which is outlined here.