Battle boats success criteria

# Introduction

Use the battle boats scenario to write the [success criteria](https://isaaccomputerscience.org/concepts/prog_cwk_analysis_aqa?topic=software_project) for your program.

* Break down the problem as much as possible into a numbered list of measurable, specific objectives
* If you have time, write success criteria for the challenge tasks as well

**Note**: add more pages if required.

|  |
| --- |
| 1 – Player must be faced with a menu to select predefined options: i.e., new game  2 – When a new game is made, display a blank battle sheet.   * + Ask user for 5 grid coordinates   + Identify if boat already exists at given coordinate   + Continually display ships after each has been placed   + Allow the user to deploy only five ships.   3 – Develop a computer-generated opponent to randomly select 5 locations to deploy the enemy ships, unknown to the player   * All the generated coordinates must be different from each other.   4 – Separate grid to allow user to target locations on the enemy's grid, begin blank  5 – Force user to take turn in order to progress game   * + Prompt user for a single set of coordinates   + Identify if there was a target at received missile location     - If target was a hit display (H)     - If target was a miss display (M)   6 – Create a subroutine for the computers turn   * + Randomly generate targeting coordinates   + Display the locked in coordinates to player   + Identify if there was a target at received missile location     - If target was a hit display (B)     - If target was a miss, record data but don’t display   7 – Develop a subroutine to progress the game until one player has lost their entire fleet. The winner should be displayed to the player.  8 – Develop a save mechanic to the game, stored externally, progress should be saved after every turn – if the game is quit progress must be resumable. Stored as a .txt file  9 – Develop a mechanic to allow resuming a game using stored data, it will resume from the last turn.  10 – The game must be able to handle errors without terminating the program   * It must be able to handle:   + Erroneous data   + Invalid data   11 – [FURTHER EXTENSION – if possible] Allow user to deploy ships of differing sizes, there will be three main classes:  2 x Destroyers (1 cell)  2 x Submarines (2 cell)  1 x Carrier (3 cells)  Boats can be placed in differing orientations – horizontally or vertically.  12 – to allow for varying boat sizes, develop subroutine to check for a hit, miss or sunken boat. The boat should only sink if all sections are compromised. |