Code Design and Data Structures

|  |  |
| --- | --- |
| **Assessment Task Number:** Part 3 – Implement a Binary Tree | |
| **Unit Code(s):** | **Unit Title(s):** |
| ICTPRG547 | Apply advanced programming skills in another language |
| CUADIG512 | Design digital applications |
| **Instructions to Learners:** | |

For this task, you must implement a binary tree and write a program that demonstrates its use.

**Create a Binary Tree Class:**

The tutorial for the session on Binary Trees walks through the creation of a Linked Binary Tree class (in which pointers are used to store the left and right branches).

For this assessment task, you may complete either the Linked Binary Tree class you started in this tutorial or create an Arrayed Binary Tree (also briefly mentioned in this tutorial).

Your Binary Tree class must include functions for inserting a new value and finding a value.

It is recommended (although not required) that you also add a function for removing a value from the tree.

**Requirements:**

The Binary Tree must support the following operations:

* Inserting a node into the binary tree
* Searching for a value

The binary tree must be ordered and maintain ordering as nodes are inserted (and removed).

Removing a node from the binary tree is recommended but not required.

**Implementation:**

There are two ways you can demonstrate the implementation of your binary tree.

1. Implement a binary space partitioning (BSP) tree within the CDDS\_Optimise program found within the AIE Student Samples solution available on GitHub ( <https://github.com/AcademyOfInteractiveEntertainment/AIEYear1Samples>)   
     
   This is one optimisation method that will significantly improve the performance of the program.

**OR**

1. Create a small, stand-alone application to demonstrate and test your binary tree. Your test application need not be a game, but it must allow the user to verify that the binary tree works without inspecting the code (an application containing a graphic user interface created using a third-party library like RayLib is recommended).

You are free to incorporate other elements or add additional features as you see fit.

|  |  |  |
| --- | --- | --- |
| **Task** | | **Evidence Criteria** |
| 1. | Binary Tree | Write a custom implementation of a binary tree and demonstrate its use in a test application. |
| **Submission Requirements:** | | |
| You will need to submit the following:   * A Release build of each application that can execute as a stand-alone program * Your complete Visual Studio project   Be sure to remove any temporary build folders (i.e., the Debug and Release folders). Only project files, source code files, and any resource files used should be included in your submission.  Package all files in a single compressed archive file (.zip, .7z, or .rar) | | |