**ASSIGNMENT BRIEF**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Module Code** | 2CB101 | **Module Tutor** | | Dr Andrew Guest | | | |
| **Module Title** | Programming 03 | | | | | | |
| **Level** | 5 | **Credit Value of Module** | | | 20 | | |
| **Assessment Task** | Software development and documentation  Essay/comparative analysis | | | | | | |
| **Word Count/Time** | N/A | | | | | | |
| **Assessment No** | 1 | **of** | 1 | | | **Weighting** | 100% |
| **Type of Submission** | Code and documentation | | | | | | |
| **Method of Submission** | Electronic through moodle & repl.it | | | | | | |
| **Publication Date** | 12/10/2020 | | | | | | |
| **Due Date** | 12/01/2021 | | | | | | |
| **Expected Feedback Date** | 2/2/2021 | | | | | | |
| **Format of Feedback** | Through moodle | | | | | | |
| **Module Learning Outcomes** | | | | | | | |
| 1. Demonstrate understanding of facts, concepts, principles and theories relating to intermediate level computer programming; 2. analyse, model, design, test, and evaluate intermediate level computer programmes; 3. apply appropriate theory, practices and tools for the specification, design, development and evaluation of intermediate computing systems including programming in a high level language;   Solve problems and represent ideas at different levels of abstraction. | | | | | | | |
| **Assignment Description** | | | | | | | |
| A Brand New Pokedex You’ve been tasked with developing a prototype for a new Pokedex that uses some form of linked list to store the player and Pokemon data. Task Summary Develop a program that mimics a Pokedex from Pokemon. A Pokedex is an encyclopedia type information system that stores a list of Pokemon and information about those Pokemon. (see [www.Pokemon.com/uk/Pokedex/](http://www.pokemon.com/uk/pokedex/)).  You need to implement the following   1. A Struct to represent the top level of the Pokedex. It holds the head of two lists – a list of players and a list of Pokemon. 2. A list of Pokemon, storing the details of a Pokemon in each node. 3. A list of Players. Each node represents a player and stores a list of Pokemon captured by that player. 4. Pokedex Functionality 5. [Optional for bonus marks] Implement optional Pokedex functionality   You also need to write a report on your program that covers   1. The Structs you have created 2. The list types you chose and why 3. Where in memory the various parts of the Pokedex are stored, and what this means. 4. Explain PlayersPokemon() function – how it works. 5. [Optional for bonus marks] Explain how optional functionality works.  Pokedex ProgramPokedex Structure You should create a struct called *Pokedex* which stored two pointers – one to the head of the player list and one to the head of the Pokemon list. A variable of the Pokedex type called *pokedex* should be created. You should pass this *pokedex* variable to functions rather than pointers to the head of either list. Pokemon List You should create a list of Pokemon. The list should be connected by pointers so that it can expand as required. It can be a singly or doubly linked list, or a tree structure.  Create a struct for each node in the list called PokemonNode. Each *node* in the list should store – the Pokemon’s name, its type and its primary ability as strings. Assume each Pokemon has only a single type and a single ability.  You should implement the basic functionality required for your list type, functions to – create new nodes, add a node to the list, to find a specific node (by Pokemon name).  Functions   * \*PokemonNode NewPokemonNode(name, type, ability) – creates a new node and returns a pointer to that node * void AddPokemonToList(\*pokedex, name, type, ability) – checks to see if *name* already exists in the Pokemon list. If it doesn’t it creates a new node and adds it to the list. It the Pokemon already exists it does nothing. * \*PokemonNode FindPokemon(pokedex, name) searches the Pokemon list for *name*. If it finds *name* it returns a pointer to the *name’s node* otherwise it returns NULL.  Player List You should create a list of players. The list should be connected by pointers so that it can expand as required. It can be a singly or doubly linked list, a stack, or a tree structure.  Create a struct for each node in the list called PlayerNode. Each *node* in the list should store – the player’s name as a string, a count of Pokemon owned, and a list of the Pokemon owned by that player. This can be an array or a dynamic structure. It should consist of pointers to the relevant Pokemon nodes in the Pokemon list.  You should implement the basic functionality required for your list type, functions to – create new nodes, add a node to the list, to find a specific node (by player name).  Functions   * \*PlayerNode NewPlayerNode(name) – creates a new node and returns a pointer to that node. The Pokemon count should be set to zero. * void AddPlayerToList(\*pokedex, name) – checks to see if *name* already exists in the Player list. If it doesn’t it creates a new node and adds it to the list. It the player already exists it does nothing. * \*PlayerNode FindPlayer(pokedex, name) searches the Player list for *name*. If it finds *name* it returns a pointer to the *name’s node* otherwise it returns NULL.   Additionally, you will need to implement the following functions   * void AddPokemonToPlayer(pokedex, player name, pokemon name) that adds the Pokemon to the player’s Pokemon list (if it is not already in there) and increments the Pokemon count for that player.  Pokedex Functionality You will need to implement the following functions   * void DisplayPokemonDetails(pokedex, name) – outputs the details of *name* to the screen * void DisplayPlayerDetails(pokedex, name) – outputs the details of *name* to the screen, including a list of names of all the Pokemon owned * void ListPokemon(pokedex) – outputs a list of names of all Pokemon in the Pokedex * void ListPlayers pokedex) – outputs a list of names of all players in the Pokedex  Optional Functionality You may implement any of the optional functionality bundles below for bonus marks   * Pokemon Ability Bundle   + void DisplayPokemonWithAbility(pokedex, ability) – display a list of all Pokemon **with the ability** of *ability*   + void DisplayPokemonListByAbility(pokedex) – display a list of **all** Pokemon, sorted alphabetically by ability * Pokemon Type Bundle   + void DisplayPokemonInType(pokedex, type) – display a list of all Pokemon **in the type** of *type*   + void DisplayPokemonListByType(pokedex) – display a list of **all** Pokemon, sorted alphabetically by type * Players With Bundle   + void DisplayPlayersWithPokemon(pokedex, name) – display a list of all Players who have Pokemon **with the name** *name*   + void DisplayPlayersWithAbility(pokedex, ability) – display a list of all Players who have Pokemon **with the ability** of *ability*   + void DisplayPlayersWithType(pokedex, type) – display a list of all Players who have Pokemon **in the type** of *type* * Evolution Bundle   + Update the PokemonNode structure to store a pointer to the form a Pokemon can evolve in to called *evolution*. Update NewPokemonNode() to initialise this pointer to NULL   + void AddPokemonEvolution(pokedex, pokemon, evolvedpokemon) – searches through the Pokemon list for Pokemon named *pokemon* and *evolvedpokemon*, if both exist the node for *pokemon* is updated so that its *evolution* points to the node for *evolvedpokemon*.   + void DisplayEvolutionList(pokedex) – outputs a list of all Pokemon and the form they evolve in to or None if they can’t evolve.   + void EvolvePlayersPokemon(pokedex, player, pokemon) – searches for *player* in the Player list and for *pokemon* in the Pokemon list. If both exist and *pokemon* is in *player*’s *owned* list, **and** *pokemon* can evolve then update *players owned* list so that the pointer to *pokemon* now points to the evolved version of *pokemon*.  Pokedex Report (2000 Words) Write a report describing your Pokedex application, focusing on the details below   * Describe the **struct**s you have created - Pokedex, PokemonNode and PlayerNode. Explain why you have created them the way you have * Explain what type of lists you have used for the Pokemon and Player lists. Explain why you have created then the way you have and how they work. * Explain where in memory the Pokedex variable and the Pokemon and Player lists are. Explain what this means. * Explain how *DisplayPlayerDetails(pokedex, name)* works. You should explain the functionality and also which types of memory are used. You should discuss what happens on the Stack and the Heap. * [Bonus] Bonus marks are available for explaining how any optional bundles work and how they use the memory.  Pokedex Structure Example Diagram, schematic  Description automatically generated | | | | | | | |
| **Assessment Regulations** | | | | | | | |
| * Your attention is drawn to the [University policy on cheating and plagiarism](https://www.yorksj.ac.uk/media/content-assets/registry/policies/code-of-practice-for-assessment/23.Academic_Misconduct_Policy_2020-21.pdf). Penalties will be applied where a student is found guilty of academic misconduct, including termination of programme. * You are required to [keep to the word limit set for an assessment and to note that you may be subject to penalty if you exceed that limit](https://www.yorksj.ac.uk/media/content-assets/registry/policies/code-of-practice-for-assessment/29.Agreed_Penalties_Policy_2020-21.pdf). You are required to provide an accurate word count on the cover sheet for each piece of work you submit. * [For late or non-submission of work](https://www.yorksj.ac.uk/media/content-assets/registry/policies/code-of-practice-for-assessment/29.Agreed_Penalties_Policy_2020-21.pdf) by the published deadline or an approved extended deadline, a mark of 0NS will be recorded. Where a re-assessment opportunity exists, a student will normally be permitted only one attempt to be re-assessed for a capped mark. * An extension to the published deadline may be granted to an individual student if they meet the eligibility criteria of the [Exceptional Circumstances Policy](https://www.yorksj.ac.uk/media/content-assets/registry/policies/code-of-practice-for-assessment/16.Exceptional_Circumstances_Policy_2020-21.pdf). | | | | | | | |

## Marking Guide

|  |  |  |  |
| --- | --- | --- | --- |
| **Part** | **Component** | **Overall Marks** | **Required Content** |
| **Pokedex Program (60%)** | Pokedex Structure | 5 | Struct used to hold the heads of the Player and Pokemon lists. |
| Pokemon List | 30 | * PokemonNode structure * \*PokemonNode NewPokemonNode(name, type, ability) * void AddPokemonToList(\*pokedex, name, type, ability) * \*PokemonNode FindPokemon(pokedex, name) |
|  | Player List | 40 | * PlayerNode structure * \*PlayerNode NewPlayerNode(name) * void AddPlayerToList(\*pokedex, name) * \*PlayerNode FindPlayer(pokedex, name) * void AddPokemonToPlayer(pokedex, player name, pokemon name) |
|  | Pokedex Functionality | 15 | * void DisplayPokemonDetails(pokedex, name) * void DisplayPlayerDetails(pokedex, name) * void ListPokemon(pokedex) * void ListPlayers pokedex) |
|  | Good Coding | 10 | * Good Variable names * Well Commented * Good Structure |
|  | Optional Bundles |  | Optional bundles get bonus marks. Mark is capped at 100 for the Pokedex Program. Cannot score more marks in options than for required content (i.e. the sections above) |
|  | [Optional] | 10 | Pokemon Ability Bundle   * void DisplayPokemonWithAbility(pokedex, ability) * void DisplayPokemonListByAbility(pokedex) |
|  | [Optional] | 10 | Pokemon Type Bundle   * void DisplayPokemonInType(pokedex, type) * void DisplayPokemonListByType(pokedex) |
|  | [Optional] | 15 | Players With Bundle   * void DisplayPlayersWithPokemon(pokedex, name) * void DisplayPlayersWithAbility(pokedex, ability) * void DisplayPlayersWithType(pokedex, type) |
|  | [Optional] | 20 | Evolution Bundle   * Update the PokemonNode structure * Update NewPokemonNode() * void AddPokemonEvolution(pokedex, pokemon, evolvedpokemon) * void DisplayEvolutionList(pokedex) * void EvolvePlayersPokemon(pokedex, player, pokemon) |
| **Project Report (40%) (2000 Words)** | Introduction | 5 | * Purpose, Scope, Objectives of document * Document Overview |
| Structures | 5 | Describe the Pokedex, PlayerNode and PokemonNode structures |
| Lists | 20 | Describe the Pokemon and Player lists. How you have chosen to implement them and why |
| Lists In Memory | 20 | Describe how the Pokemon and Player lists are stored in memory, how they are accessed. |
| DisplayPlayerDetails() | 35 | Explain how DisplayPlayerDetails() works in detail, paying special attention to memory locations. Discuss the Stack, the Heap and frames. |
| Conclusion | 5 | A brief conclusion summarising the report |
| Quality | 10 | Document   1. Is properly structured with headings and section numbers 2. Is readable with sentences that make sense, correct spelling and grammar 3. Is properly referenced using Harvard referencing style |
| Optional Bundles |  | Bonus marks available for explaining any optional bundles you implemented in your program. . Mark is capped at 100 for the Pokedex Report. Cannot score more marks in options than for required content (i.e. the sections above) |
|  | [Optional] | 10 | Pokemon Ability Bundle |
|  | [Optional] | 10 | Pokemon Type Bundle |
|  | [Optional] | 15 | Players With Bundle |
|  | [Optional] | 20 | Evolution Bundle |

University Generic Assessment Descriptors