ASSIGNMENT BRIEF

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Module Code | COM5001 | Module Leader | | Dr Andrew Guest | |
| Module Title | Programming 03 | | | | |
| Level | 5 | Credit Value of Module | | 20 | |
| Assessment Task | Software development and documentation | | | | |
| Word Count | 2000 words for report | | | | |
| Assessment No | 1 | of | 1 | Weighting | 100% |
| Type of Submission | Porfolio | | | | |
| Method of Submission | Electronic through moodle and gitlab | | | | |
| Publication Date | 12/10/2021 | | | | |
| Due Date | 17/01/2022 | | | | |
| Expected Feedback Date | 07/02/2022 | | | | |
| Resit Date | April 2022 | | | | |
| Format of Feedback | Through moodle | | | | |
| Anonymous marking | Not anonymous. Documentation is closely linked to code and code cannot be anonymous.  Code of Assessment 29.5.2  c) Subject-specific skills exercises or product creation e.g. artwork, product design, video production, performances. | | | | |
| 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; 4. 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   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 DisplayPlayerDetails () function – how it 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, 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 an array of pointers to** 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.   PokemonNode\* PokemonArray[20]; as part of PlayerNode, to store pointers to the nodes in the Pokemon list for the Pokemon that player has captured. 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 | | | | | |
| Pokedex Structure Example (Singly Linked List) Diagram, schematic  Description automatically generated | | | | | |
| Optional Pokedex Evolution Structure Example (Singly Linked List)Diagram, schematic  Description automatically generated | | | | | |
| **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. | | | | | |
| Assessment Regulations | | | | | |
| * Your attention is drawn to the University policy on academic misconduct ([Academic Misconduct Policy](https://www.yorksj.ac.uk/media/content-assets/registry/policies/code-of-practice-for-assessment/23.Academic_Misconduct_Policy_2021-22.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/time limit set for an assessment and to note that you may be subject to penalty if you exceed that limit ([Agreed Penalties Policy](https://www.yorksj.ac.uk/media/content-assets/registry/policies/code-of-practice-for-assessment/30.Agreed_Penalties_Policy_2021-22.pdf)). You are required to provide an accurate word count on the cover sheet for each piece of work you submit. * For a first assessment attempt a penalty may be applied for late or non-submission of work by the published deadline or an approved extended deadline ([Agreed Penalties Policy](https://www.yorksj.ac.uk/media/content-assets/registry/policies/code-of-practice-for-assessment/30.Agreed_Penalties_Policy_2021-22.pdf)). * Where a re-assessment opportunity exists, late or non-submission of work receives a mark of zero and is not eligible for a capped mark ([Agreed Penalties Policy](https://www.yorksj.ac.uk/media/content-assets/registry/policies/code-of-practice-for-assessment/30.Agreed_Penalties_Policy_2021-22.pdf) and [Reassessment](https://www.yorksj.ac.uk/students/exams-and-assessment/reassessment/)) * 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_2021-22.pdf). | | | | | |
| Note | | | | | |
| [Feedback Policy:](https://www.yorksj.ac.uk/media/content-assets/registry/policies/code-of-practice-for-assessment/31.Feedback_Policy_2021-22.pdf) Marks are to be returned to students with the caveat that all marks are provisional until final approval by the School Assessment Board (SAB). Confirmed marks will be made available via e:Vision after the SAB meeting. | | | | | |

**Marking Rubric**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Code - Player List [15%]** | [85%+] Data structures correct. All functions implemented fully. Fully commented | [70-85%] Data structures correct. All functions implemented. | [60-69%] Data structures correct. Most functions implemented. | [50-59%] Data structures correct. Some functions implemented. All functions attempted. | [40-49%] Data structures correct. Some functions implemented. | [20-39%]Data structures and functions not implemented correctly but a valid attempt was made. | [0-19%]Little evidence of valid attempt to code the player list. |
| ***25points*** | ***21points*** | ***18points*** | ***15points*** | ***12points*** | ***6points*** | ***0points*** |
| **Code - Pokemon List [20%]** | [85%+] Data structures correct. All functions implemented fully. Fully commented | [70-85%] Data structures correct. All functions implemented. | [60-69%] Data structures correct. Most functions implemented. | [50-59%] Data structures correct. Some functions implemented. All functions attempted. | [40-49%]Data structures correct. Some functions implemented. | [20-39%]Data structures and functions not implemented correctly but a valid attempt was made. | [0-19%]Little evidence of valid attempt to code the pokemon list. |
| ***34points*** | ***28points*** | ***24points*** | ***20points*** | ***16points*** | ***8points*** | ***0points*** |
| **Code - Pokedex Functionality [10%]** | [85%+] All functions implemented fully. Fully commented | [70-85%] All functions implemented. | [60-69%] Most functions implemented. | [50-59%] Some functions implemented. All functions attempted. | [40-49%]Some functions implemented. | [20-39%]Functions not implemented correctly but a valid attempt was made. | [0-19%]Little evidence of valid attempt to code the pokedex functionality. |
| ***17points*** | ***14points*** | ***12points*** | ***10points*** | ***8points*** | ***4points*** | ***0points*** |
| **Code - Coding Style [5%]** | [85%+] Code is well structured. Broken down in to appropriate files. Functions and variables well named. | [70-85%] Code is well structured. Functions and variables well named. | [60-69%] Code has decent structure but could be better. Variables and functions are mostly well named. | [50-59%] Code has some structure but could be much better. Variables and functions are named okay. | [40-49%]Code structure is poor, barely acceptable. Some variables and functions have reasonable names. | [20-39%]Code has barely any structure. Variables and functions are poorly named. | [0-19%]Little attempt made to structure code. Variables & functions are seldmon named appropriately. |
| ***9points*** | ***7points*** | ***6points*** | ***5points*** | ***4points*** | ***2points*** | ***0points*** |
| **Report - Lists [12.5%]** | [85%+] The Pokedex structure is described well and clearly. Choices are explained well. Good diagrams used. | [70-85%] The Pokedex structure is described well. Choices are explained well. Appropriate diagrams used. | [60-69%] The Pokedex structure is described. Choices are explained. Diagrams used. | [50-59%] The basic Pokedex structure is described. Choices are explained. Some diagrams used. | [40-49%]The Pokedex structure is described briefly. Choices are not explained. No diagrams used. | [20-39%]Pokedex structure poorly described. Choices not explained. | [0-19%]Pokedex structure description is very poor. |
| ***22points*** | ***18points*** | ***15points*** | ***13points*** | ***10points*** | ***5points*** | ***0points*** |
| **Report - Memory Usage [12.5%]** | [85%+] Where each component of the Pokedex is stored in memory and how they are accessed is described excellently. Good diagrams used. | [70-85%] Where each component of the Pokedex is stored in memory and how they are accessed is described well. Good diagrams used. | [60-69%] Where each component of the Pokedex is stored in memory and how they are accessed is described . Good diagrams used. | [50-59%] Where most components of the Pokedex is stored in memory and how they are accesseddescribed. Diagrams used. | [40-49%]Where some components of the Pokedex are stored in memory and how they are accessed is described. Diagrams used innefectively.. | [20-39%]Poor attempt at describing where components of the Pokedex are stored in memory or accessed is described. | [0-19%]No attempt to descibe how the pokedex is stored in memory or accessed. |
| ***22points*** | ***18points*** | ***15points*** | ***13points*** | ***10points*** | ***5points*** | ***0points*** |
| **Report - DisplayPlayerDetails() [17.5%]** | Function described well and in detail. How the memory is used and when is also described well. Good diagrams used. | [70-85%] Function described well . How the memory is used and when is also described. Good diagrams used. | [60-69%] Function described well . How the memory is used or when is described. Diagrams used. | [50-59%] Function described. How the memory is used or when is described. Adequate diagrams used. | [40-49%]Function described poorly. How the memory is used or when is described. | [20-39%]Function described poorly. | [0-19%]Function barely described. |
| ***30points*** | ***25points*** | ***21points*** | ***18points*** | ***14points*** | ***7points*** | ***0points*** |
| **Report - Structure & Style [7.5%]** | [85%+] Report structure excellent. Writing style excellent. | [70-85%] Report structure very good. Writing style very good. | [60-69%] Report structure good. Writing style good. Spelling & Punctuation mostly good. | [50-59%] Report structure okay. Writing style okay. Possibly spelling & punctuation errors. | [40-49%]Report structure okay. Writing style okay. Frequent spelling & punctuation errors. Some parts hard to follow. | [20-39%]Report structure poor. Writing style poor. Many spelling & punctuation errors. Some parts hard to follow. | [0-19%]Report structure very poor. Writing style very poor. Many spelling & punctuation errors. Some parts hard to follow. |
| ***13points*** | ***11points*** | ***9points*** | ***8points*** | ***6points*** | ***3points*** | ***0points*** |