|  |  |
| --- | --- |
| **Task** | **Point** |
| Structure Chart | 6 |
| Data Dictionary | 3 |
| P-spec | 5 |
| Presentation | 2 |
| **GROUP TOTAL** | **16.0** |
| Individual Paragraph for ***John, Alex, Erik, Logan, Steven*** | 2 |
| **TOTAL** | **18** |

* ***Data dictionary document the data items. Do not include the subroutines or functions in the data dictionary. Also, you need to read the requirement for this deliverable carefully. The requirement is to include the following information in the data dictionary:***
  + - *Name of the data item*
    - *Description of how the data item is used*
    - *Data type*
    - *Initial value*
    - *Range of values*
    - *The functions/modules using/referencing the data item*

**Structured Analysis – Nibbles**

*1337 hAXX – John Polus, Alex Jacobs, Erik Rasmussen, Logan Brincks, Steven Karrmann*

1. **Introduction**

The game that we are re-engineering for the second project is Nibbles. Nibbles similar to snake, where the player guides the character and tries to eat numbers that appear on the screen without running into his own tail or the wall. If the player collides with their own tail, the game ends. The game was written in the language BASIC in 1991 and published by Microsoft. The source code consists of a single file Nibble.BAS.

1. **Structured Analysis**

**2.1 Structure Chart**

The source code contains 14 subprocedure and 2 functions. Due to the structure diagram being too large, it could not be included in this document. For the structure chart of Nibbles.BAS, see Structure Diagram.docx in the same directory as this document.

**2.2 Data Dictionary**

Like the structure chart, the data dictionary was too large to include here. For the data dictionary of Nibbles.BAS, please see DataDictionary.doc in the same directory as this document.

**2.3 P-Spec**

The game Nibbles contains 14 subprograms and 2 functions. After analysis and the creation of the corresponding P-spec tables below, it was discovered that there were subprocedures undefined and inconsistent naming conventions making interpretation of functionality difficult. Furthermore, most of the subprocedures are involved in either graphical display or text display. Therefore, there is only a moderate amount of data modification within subprocedure.

It is worth mentioning that due to clarity issues in regard to certain naming conventions; complete understanding of some subprocedure functionality may still be somewhat unclear. The P-spec tables for Nibbles.BAS are as follows:

|  |
| --- |
| Process Name: SpacePause |
| Description: Pauses game play and waits for space bar to be pressed before continuing. Also, displays a message to the screen, which is erased before game plays starts again. |
| Pre-condition: Player has pressed the space bar |
| Post-condition: Gameplay has started again |

|  |
| --- |
| Process Name: PrintScore |
| Description: Prints player’s scores and number of lives remaining. |
| Pre-condition: Game has started |
| Post-condition: Current score and number of lives has been printed to the screen |

|  |
| --- |
| Process Name: Intro |
| Description: Displays game introduction, which includes controls and instructions. |
| Pre-condition: A game has not yet started |
| Post-condition: Text has been written to the screen |

|  |
| --- |
| Process Name: GetInputs |
| Description: Gets player inputs from the keyboard. |
| Pre-condition: Player has chosen to start the game |
| Post-condition: Settings for the current game have been chosen, such as difficulty level, number of players, and game speed |

|  |
| --- |
| Process Name: DrawScreen |
| Description: Draws the playing field for the game. |
| Pre-condition: Game has started (called every frame) |
| Post-condition: Field has been drawn to the screen |

|  |
| --- |
| Process Name: PlayNibbles |
| Description: Main routine that controls game play. |
| Pre-condition: Player has chosen to start the game and selected the appropriate settings |
| Post-condition: Player has run out of lives or chosen to quit |

|  |
| --- |
| Process Name: Set |
| Description: Sets row and column on playing field to given color to facilitate moving of snakes around the field. |
| Pre-condition: Game has started (called every frame) |
| Post-condition: Color of square at the given coordinates is the new color |

|  |
| --- |
| Process Name: Center |
| Description: Centers text on given row. |
| Pre-condition: Text to be displayed has been generated |
| Post-condition: Text has been written to the screen with equal empty space on to its left and right |

|  |
| --- |
| Process Name: DoIntro |
| Description: Unimplemented. |
| Pre-condition: Unimplemented |
| Post-condition: Unimplemented |

|  |
| --- |
| Process Name: Initialize |
| Description: Unimplemented. |
| Pre-condition: Unimplemented |
| Post-condition: Unimplemented |

|  |
| --- |
| Process Name: SparklePause |
| Description: Creates flashing border for intro screen. |
| Pre-condition: Player has not chosen to start the game |
| Post-condition: “Sparkles” have been drawn to the screen |

|  |
| --- |
| Process Name: Level |
| Description: Sets game level according to WhatToDO and updates the snake for the start of that level. |
| Pre-condition: Level has ended or game is just starting |
| Post-condition: Next level has started |

|  |
| --- |
| Process Name: InitColors |
| Description: Initializes playing field colors at the beginning of the game. |
| Pre-condition: Player has chosen to start the game |
| Post-condition: Playing field colors have been initialized |

|  |
| --- |
| Process Name: EraseSnake |
| Description: Erases snake to facilitate moving through playing field. |
| Pre-condition: Game has started (called every frame) |
| Post-condition: Part of snake that should be gone after each frame is erased |

|  |
| --- |
| Process Name: StillWantsToPlay |
| Description: Determines if user wants to play, game again based on user input. |
| Pre-condition: Player has run out of lives |
| Post-condition: New game is started or program ends |

|  |
| --- |
| Process Name: PointIsThere |
| Description: Checks the global arena array to see if the boolean flag for the given point is set. |
| Pre-condition: Game has started (called every frame) |
| Post-condition: Boolean value has been returned |

**2.4 Software Metrics**

Lines of Code (LoC): 539

Comment Lines: 77

Total Lines (Including Whitespace): 616 (722)

Functions: 2

Subprocedures: 14 (2 unimplemented)