**Test Plan for Problem 1 A**

**Checkpoints**

|  |  |
| --- | --- |
| Inputs several lines of text from the keyboard | Y / N |
| Prints alphabetical listing of each word from the input | Y / N |
| Shows the frequency of each word | Y / N |
| Prints the name and description of the program | Y / N |
| Uses qsort to sort the words | Y / N |
| Ignores case differences | Y / N |
| Input is terminated by the user entering the EOF character | Y / N |
| One or more spaces separate the words | Y / N |
| Each word is shown only once | Y / N |
| Groups words by first letter | Y / N |

**Test data**

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| Two lines of text. Words separated by spaces. ^Z is entered on the last line to stop input. | The two lines of text should be inputted and the words are alphabetically sorted and grouped. Also, the word frequency is shown. |  |
| One line of text is inputted, ^Z is entered on the last line to stop the input. | The line of text should be inputted and the words are alphabetically sorted and grouped. Also, the word frequency is shown. |  |

**Test Plan for Problem 1 B**

**Checkpoints**

|  |  |
| --- | --- |
| The program prints out the title and description | Y / N |
| Passes all the checkpoints in part A, except for the EOF checkpoint | Y / N |
| The program allows a user to enter a file name to read | Y / N |
| If the file does not exist the program exits | Y / N |

**Test data**

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| A file named test.txt is created and multiple lines of text are placed into it. Then the file name is entered. | The words in the file test.txt is displayed alphabetically. Also, the frequency of each word is shown. |  |
| A file that does not exist is entered. | The program tells the user that the program is exiting and then the program exits. |  |

**Test Plan for Problem 2**

**Checkpoints**

|  |  |
| --- | --- |
| Program prints purpose of the program, title, and menu | Y / N |
| Allows the user to modify any data pertaining to the records | Y / N |
| If a file is the file hardware.dat does not exit, the user is given the option to create a new hardware.dat file that it initialized to the records found on the assignment sheet | Y / N |
| The user can create new records | Y / N |
| The user can delete all records | Y / N |
| The records include the data: record#, tool name, quantity, and cost. | Y / N |
| The user can save the modifications to the data in the file hardware.dat | Y / N |
| Input is validated | Y / N |
|  |  |

**Test data**

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| Y | Creates a file hardware.dat with the data in the assignment sheet. |  |
| 1 | Lists the contents of hardware.dat |  |
| 2 3 | If record 3 exists, then the edit menu is shown. If not, the user is warned and prompted to enter in a y or n to continue. |  |
| Test all menu options. Meaning the input will be: 1, 1 2, etc. All menu paths will be tested. | They all work |  |

**Test Plan for Problem 3 A**

**Checkpoints**

|  |  |
| --- | --- |
| Program prints purpose of the program and title | Y / N |
| The gymMember structure holds data pertaining to: name, height, date of joining, and fitness score | Y / N |
| The fitness score contains values on the range of [0,100] | Y / N |
| The << operator is overloaded for the gymMember structure and is placed in the file gymMember.cpp | Y / N |
| The program title and description is displayed | Y / N |
| The record data is outputted after the user fills up the array of gymMember structures | Y / N |
| The output is printed out in a well formatted table | Y / N |
| Data is validated | Y / N |
| **Test data**   |  |  |  | | --- | --- | --- | | Input | Expected Output | Actual Output | | Andrew 21 5.11 12 13 1989 | Stores the data into record one and prompts for the next records data. |  | |  |
|  |  |

**Test Plan for Problem 3 B**

**Checkpoints**

|  |  |
| --- | --- |
| Passes the same checkpoints and tests as plan 3A. | Y / N |
| Uses an dynamic array instead of a static array as used in problem 3A | Y / N |
| **Test data**   |  |  |  | | --- | --- | --- | | Input | Expected Output | Actual Output | | 1 Andrew 21 5.11 12 13 1989 | Stores the data into a record and prints out the record data in a nicely formatted table. |  | |
|  |

**Test Plan for Problem 4**

**Checkpoints**

|  |  |
| --- | --- |
| Program prints purpose of the program and title | Y / N |
| The gymMember structure holds data pertaining to: name, height, date of joining, and fitness score | Y / N |
| The fitness score contains values on the range of [0,100] | Y / N |
| The << operator is overloaded for the gymMember structure and is placed in the file gymMember.cpp | Y / N |
| The program title and description is displayed | Y / N |
| The program formats the output into a nice table | Y / N |
| At least one of the gymMember elements fields are initialized by the overloaded constructor | Y / N |
|  |  |

**Test data**

There is no test data, for the tests are hardcoded.