Bobcat Aerospace
Software Requirements Document

Software Name: Craft Designer Plus
Author: Gentry Atkinson
Version: 2.0.0

1. Introduction
Product Purpose: this product is being developed for the purpose of facilitating space vehicle design. This software will be able to take input from an engineer or designer and generate simple statistics about one vehicle which will be saved in a text file. The product will also be able to test these design files against some simple parameters.

2. Standards
Hardware: this product will be capable of running on any platform with a C++ compiler and a means of text I/O.

Schedule: completion of this product is expected no later than June 16.

Language: this project will be implemented using the C++ programming language

3. System Description

System Context: this will be a self-contained piece of software that will not rely on external platforms for its basic operation.

User Characteristics: users will be trained aerospace designers who are proficient with computer operation. This project will not be responsible for user training.

4. Functional Requirements

Menu: the user should be shown a simple text interface with the following options:

- **D**- design a new ship
- **T**- test a new ship
- **Q**- quit the program

This menu should allow the user to select one option and should print an error message if an invalid option is chosen. Based on user input one of the following subroutines should be run. Each subroutine should return to the menu after running, except for '**Q**'. The user should be able to enter a capital or lowercase letter as their input.

4.1 Design a New Ship (same as last program)

Input: the product will user collect the following information from the user through a simple text interface:

- **Designer Name**: the first and last name of the vehicle's designer
- **Ship Name:** a one word name for the vehicle
- **Ship Mass:** a numerical value representing the mass of a vehicle in kilograms
- **Engine Thrust:** a numerical value representing the force exerted by the vehicle's engines in newtons.

Output: the product should write the following values to a text file:

- **Ship Name**
- **Ship Mass**
- **Acceleration:** calculated using (**Engine Thrust/Ship Mass**)
- **Impulse:** calculated as **g0 * ln(Acceleration),** where **g0** is the constant gravitation value 9.8 and **ln** is the natural log function.
- **Maximum Altitude:** calculated as **(100 + (Impulse*200))/10**, with the result being the kilometers above sea level that the vehicle can safely travel.
- Designer Name

Interface: the product should format the output as follows:

- "File written for [Ship Name]" should be printed on the **console** with [Ship Name] replaced by the actual Ship Name.
- The program's output should be written to a **file** named [Ship Name].txt with [Ship Name] replaced by the actual Ship Name.
- The file output should be formatted as follows:

```
##########[Ship Name]##########      //10 #s before and after
                                     //blank line
Ship mass:          [Ship Mass]kg    //Values in neat columns
Acceleration:       [Acceleration]m/s2 //
Impulse:            [Impulse]Ns/kg   //
Max Altitude:       [Max Alt.]km     //
                                     //blank line
File saved at:      [time stamp]     //
Designed by:        [Designer Name]  //
```

  - All [Values] should be replace with correct values

- Comments should not appear in the file.
- Every item of output should have the correct unit as shown.

4.2 Test a Ship Design

Input: the product will prompt the user for the name of one space vehicle.

Output: the product will open one stored ship design file using the user's input as the file name. If that file does not exist the product will print an error message and return to the menu. Otherwise, all values should be read from the file and the following tests performed:

- **Test 1:** the Designer Name should not be "Thomas Harris". He was fired.
- **Test 2**: the ship mass should be less than 1000kg
- **Test 3**: the Acceleration should be more than 1 m/s2 but less than 10 m/s2
- **Test 4**: the Impulse must be at least 10.0 Ns/kg
- **Test 5:** if the Ship Mass is less than 100kg, then the Max Altitude must be at least 150km. Otherwise the Max Altitude must be at least 200km.

Interface: any ship that fails one of the tests should have the number of the first test that it fails printed to the console in this format:

[Ship Name] has failed test number [Test Number]

A ship that passes all of the tests should have the following message printed to the console:

[Ship Name] has passed all tests.

The output should have [Ship Name] and [Test Number] replaced with the appropriate values.

4.3 Quit

Input: the product will take no input

Output: the product will print a parting message on the console.

Interface: the parting message should "Thank you for using Craft Designer Plus".

5. Non-functional Requirements

- The program should be submitted as a .cpp file with the name [Author's Last Name]_assignment2.cpp, with [Author's Last Name] replaced by the author's last name.
- The first two lines of the program should be the author's name and the date that the assignment was written.
- The author is expected to follow the Style Guidelines.
- Numbers should be formatted to one decimal place of accuracy.

6. Sample Test Cases:

Input
```
d
Robert Goddard
Zoomer1
1000
10000
```
Console
```
File written for Zoomer1
```
Zoomer1.txt
```
##########Zoomer1##########

Ship Mass:          1000.0kg
Acceleration:       10.0m/s2
Impulse:            22.6Ns/kg
Max Altitude:       461.3km

File saved at:      1620926125
Deigned By:         Robert Goddard
```

Input
```
d
Sergei Korolev
Boko
200
250
```
Console
```
File written for Boko
```
Boko.txt
```
##########Boko##########

Ship Mass:          200.0kg
Acceleration:       1.2m/s2
Impulse:            2.2Ns/kg
Max Altitude:       53.7km

File saved at:      1620956317
Designed by:        Sergei Korolev
```

Input
```
T
Zoomer1
```
Console
```
Zoomer1 has failed Test 2
```

Input
```
T
Boko
```
Console
```
Boko has failed Test 4.
```

7. Guidance

- The **<iomanip>** library can be used to set the precision of decimal numbers using the **fixed** and **setprecision** keywords.
- The **log** function from the **<cmath>** library can be used to calculate a natural log.
- The function **time(NULL)** from the **<ctime>** can be used to produce a timestamp formatted in computer time. Printing a timestamp on the screen would look like:
  - **cout << time(NULL) << endl;**
- When reading a text file, you will need to discard some string values.
- You will need to use a loop to keep the program running until the user enters ‘q’ or ‘Q’ as a menu option.