COMP 2006L – Formula Calculator & Integrator

User Requirements for the Project

June 1, 2015

1 Introduction

Commercial software development often begins with a process of *requirements gathering*. The clients who request the software have certain expectations and requirements that should make it into the final product in some form. **User Requirements** are thus the set of features and behaviours that our finished software should provide to the end-user of the developed system.

User Requirements are elicited from end-users and other stakeholders during a phase of consultation. These requirements may be modified or interpreted in specific ways by the development team. Generally speaking, a development team should obtain buy-in from these stakeholders before proceeding with their vision of the finished product. For this reason the development of an early prototype system, in which the stakeholders can see the developers' vision in concrete terms, is highly recommended.
For this project you may consider me to be the main stakeholder: as the development team it is my requirements for the system that you should interpret and develop. This document lays out my user requirements for the project. A degree of wiggle room is provided for you to interpret these requirements in your own way, and provided you make a good faith effort to implement each of the requirements, your particular vision will not be penalized if it does not conform with mine.
Your interim report will provide you with an opportunity to lay out your vision of the project, though will be free to change your design after submitting this report.
Your final report will provide you with an opportunity to explain and justify the design decision you have taken in the implementation of these requirements. Your vision of the project need not be identical to mine or to that of your classmates. It is sufficient that you make a good faith attempt to implement the requirements and that you are justify your actual implementation decisions in your final report.
Projects will be graded for the fullness of their implementation with respect to the user requirements provided here, as well as for any aspects that make it exemplary or unique in design, conception or execution. Remember to document any special features that you may implement, while being honest about those that you overlook or under-develop.

2 FormCalc User Requirements

FormCalc is a symbolic calculator that allows users to manipulate real mathematical formulae in their textual/symbolic forms. You may think of **FormCalc** as a cross between a standard calculator app on your desktop and the Mathematica software package.
In class we have incrementally developed the core representations that will underpin the FormCalc system. You will thus already have the necessary classes for representing a complex symbolic formula in a natural OOP fashion, and the necessary methods for converting a flat-string formula into this representation, as well as the necessary methods for generating a corresponding printable/displayable flat-string output from these representations.

Requirement 1: User Interface

Your interface will comprise two components:

- **Requirement 1a: Text User Interface:** A user may use a text window to communicate with FormCalc, for example, to enter formulae, evaluate formulae, differentiate formula, request a graph of formulae, etc. The text interface scrolls, to show previous commands that were entered.
- **Requirement 1b: Graphical "calculator-style" User Interface:** A user may also use a graphical calculator interface – with buttons and readout strip – to interact with FormCalc and thereby enter expressions to be evaluated or graphed. Check out a standard calculator app (such as the one that comes bundled with Windows or Mac OS, or the one on your smart-phone) for ideas.

Requirement 2: Formula entry and storage

A user may enter a formula (as complex as he/she likes) through the text interface, and assign a given name to this formula (e.g., f, g, etc.). **FormCalc** will remember any named formula it is given, for possible use in future formulae.
If a formula is assigned a name that is already in use, **FormCalc** will ask the user whether it should over-write this name and associate it with the new formula.

Requirement 3: Formula re-use

A user may use an existing named formula as a function in another formula. For instance, if the user has assigned the formula $x^2 + 7$ to the function name f, then it is possible to use terms like $f(x)$ or $f(22)$ in another formula, or at the text interface to be evaluated. For instance, a user may later assign the name g to the formula $f(x^2)/x$.

Requirement 4: Formula Evaluation

If a user has given a name to a given formula, and that formula contains one of more variables (let's call them x, y and z), then the user can evaluate a formula for specific values of these variables as follows:

- **Requirement 4a: One variable formulae:** When a formula contains just one variable, that formula can be evaluated for a specific value of that variable by writing the name of the formula followed by a value in parentheses, as in $f(7)$ or $g(22)$.
- **Requirement 4b: multi-variable formulae** When a formula contains multiple variables, that formula can be evaluated for a specific value of that variable by writing the name of the formula followed by a sequence of value assignments in parentheses, as in $f(x = 7,\ \ y = 5)$ or $g(x = 6,\ \ y = 2,\ \ z = 1)$.
- **Requirement 4c: evaluating nested formulae** The value assigned to a formula variable in (4a) or (4b) above may be a constant or it may be an expression, as in $f(g(7))$ or $g(x = f(6),\ \ y = f(2),\ \ z = f(9))$.

Requirement 5: Formula Integration

The integral function of a formula is denoted by the name of the formula in uppercase. So the integral function of $f()$ is $F()$ or $\int f()$.
FormCalc can calculate the integral of any formula $f()$ using numerical method (which we do later in class and which you will implement as a weekly assignment).
FormCalc should implement an integration module also, and associate the formula for the integral of $f()$ with the name $F()$ or $\int f()$. The rules of integration (which you will have learned in high-school, or in 1st year of college) are applied, to generate a new formula from a given formula.
For instance, if $f(x) = 2x + 2$ then $F(x) = \int f(x)dx = x^2 + 2x + c$.

Requirement 6: Formula Graphing

FormCalc can graph any named formula of one free variable that it is given. It will do so in its own GUI window/canvas, which will be labeled with the name of the formula.
To ask **FormCalc** to graph a formula $f()$ (say), the user may type something like the following into the text interface:
graph $f(x = 0,\ 100)$
Notice the use of equals and comma here. The argument $x = 0, 100$ tells **FormCalc** that the value for the variable x should run from 0 to 100 and that the corresponding value of $f(x)$ should be calculated for each x and graphed accordingly.
How should x be incremented as it runs between these two end-values? If no increment is given, **FormCalc** should choose its own. However, one may be given as follows:

$$\text{graph } f(x = 0,\ 100,\ 1) \tag{1}$$

Here the value 1 is explicitly provided as an increment. **FormCalc** should thus vary x from 0 to 100 in 100 increments of 1, and evaluate $f(x)$ for all 101 values of x in [0..100].
A multi-variable formula can also be graphed if only one of its variables is free to change. We might thus ask **FormCalc** to graph the formula $g()$ as follows:

$$\text{graph } g(x = 0, 100, 1 \quad y = 2 \quad z = 9) \tag{2}$$

Though g is a function of three variables, two of them are held at constant values (2 and 9) while only x is free to change over the interval [0..100] in increments of 1.
Naturally, **FormCalc** can also be asked to graph the integral of a formula as follows:

$$\text{graph } \int_{x=0}^{100} g(x, y, z) \qquad \text{for } \ y = 2 \quad z = 9 \tag{3}$$

Requirement 7: Graphing Multiple Formula simultaneously

FormCalc can be asked to graph multiple formula in the same window. Simply, the sur specifies the different formulae on the same graph command, as in this example:

$$\text{graph } f(x = 0, 100, 1) \ \ and \ \ \int_{x=0}^{100} f(x) \tag{4}$$

This command will draw $f()$ and its integral on top of each other in a window.

Requirement 8: Saving Formulae to a file

FormCalc can be asked to save its memory of formulae to a disk file. This functionality can be initiated by typing the command save into the text interface or by pressing a save button in the GUI interface. A file browser window will open in either case, allowing the user to specify a filename into which the formulae should be saved. The formulae will then be saved in a text file that may be read by a human using a text editor or a browser. Consider using an XML format for your file (though plain text is acceptable).

Requirement 9: Re-Loading Formulae from a file

FormCalc can be asked to re-load a set of formulae from a disk file. This functionality can be initiated by typing the command load into the text interface or by pressing a load button in the GUI interface. A file browser window will open in either case, allowing the user to specify the file from which the formulae should be loaded. The file (as in 8 above) will associate formulae with their given names, and these names will be preserved when the file is re-loaded. Thus, if a user defines a formula for $f()$ and $g()$ and then saves these, a later reload will allow FormCalc to remember the definitions of $f()$ and $g()$.
When reloading a file containing named formula (as e.g. saved in 8) FormCalc may load a formula for a name that is already in use with a different formula. In this case, **FormCalc** should alert the user and ask for permission to overwrite the definition in memory with the definition on file.

Requirement 10: Saving a graph to a file

FormCalc can be asked to save a graph to disk. This functionality can be initiated by typing the command save graph into the text interface as shown below:

$$\text{save graph} \quad F(x = 0, 100, 1) \tag{5}$$

For this input, **FormCalc** will calculate the points in the correspond graph and write them out to disk, in a comma-separated-values (.CSV) file-format. In the above example, it will calculate $F(x)$ for all integer values of x in the interval [0..100] and write out to disk the value of x and the value of $F(x)$ on the same comma-separated-values line.
To find out which file to write to, **FormCalc** will open a file-browser window as in 9 above.

Requirement 11: Find area under curve

The area under a curve in a graph $f(x)$ can be calculated by solving the definite integral where a and b are the start and end points for x. FormCalc should provide a function to calculate this.