Easy Math: A software package for learning Basic Arithmetic

by
Charles Betow

Problem Report Submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Computer Science

James D. Mooney, Ph.D, Chair
Don Adjeroh, Ph.D.
Afzel Noore, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, WV
2008

ABSTRACT

Easy Math: A software package for learning Basic Arithmetic

Charles O. Betow

The use of computer hardware and software in education and training dates to the early 1940s, when American researchers developed flight simulators which used analog computers to generate simulated on board instrument data. However, there were some limitations to the developmental progress of educational software since their development was directly tied to the mainframe computers on which they ran. However, mainframe computers were not commonplace at that time. The arrival of the personal computer in 1975 changed the software field in general and even more so the development of educational software. Prior to 1975, users were dependent upon university or government owned main frames. By the early 1980s, the availability of the personal computer allowed for the creation of companies and non profit organizations which specialized in the development and distribution of educational software. The unbounded growth in the development of software geared primarily towards children's education actually occurred in the 1990s when thousands of software packages were released all focused on different aspects in the education for children. The design of these software packages for children has been very strongly influenced by computer gaming concepts. They are designed to capture the children's attention as well as educating them thus the developers ensure that the entertainment factor of these packages do not outweigh the educational factor.

The results of research on evaluating educational software packages have led technology education scholars to conclude that, instructional software packages for children should have five main functions: *drill and practice, tutorials, simulation, instructional game and problem solving.* The different available packages in the software market for children should touch on at least one of these five main functions. Easy Math has been designed to exploit the functions of ***drill and practice***[1], ***problem solving***, as well as ***tutorials***[2]. Easy math is a classic example of technology integration to facilitate the learning and teaching process. Easy Math software provides short notes and tips for basic arithmetic (addition, subtraction, multiplication, division and working with fractions), and also provides practice tests to help assess the users' progress. This package will serve as a complement and not a replacement of the class teacher. It should allow the user to improve their basic arithmetic skills.

[1] Drill and Practice software provides exercises in which students work example items and receive feedback on their correctness.

[2] An entire instructional sequence similar to a teacher's classroom instruction on a topic.

Table of Contents:

Chapter 1

Introduction

The objective of this project is to develop a software application called Easy Math. Easy Math is a highly customized non marketable ***Instructional software*** [3]package created specifically to meet the needs of ***Aunt Hattie's place (AHP)*** (a home for underprivileged young black males in the greater Maryland area). The software has been created to help them improve their basic arithmetic skills such as addition, subtraction, multiplication, division, and the manipulation of signed numbers and fractions.

1.1 Hattie's Place (AHP)

Aunt Hattie's Place (AHP) is a non-profit organization dedicated to providing a safe, stable, nurturing, and long-term home for abused, abandoned, and neglected children in foster care. Founded in February, 1997 by Dr. Hattie N. Washington, AHP strives to provide a stable home for youngsters who will be in foster care long-term--often their entire childhood and teenage years. AHP is licensed by the Department of Human Resources; for licensing purposes AHP is considered a group home and is a 501(c) (3) non-profit corporation.

AHP currently provides a home to eighteen young boys, 7-18 years old. These children, lovingly called "SuperKids", come to AHP with a wide variety of psychological and behavioral problems, as well as poor academic records. They usually have a sense of hopelessness, lack of respect for authority, and a general negative outlook on their future. These problems typically are associated with feelings of loss and anger resulting from

[3] Software that allow students to learn new content, practice using content already learned and/or be evaluated on how much they know

their parents' vicious cycle of substance abuse and/or incarceration, and are exacerbated by the stigma identified with being in foster care.

AHP (http://www.aunthattiesplace.org/ahp_index.html) has been cited by the Department of Human Resources as a model for other group homes, because of the significant progress its residents have made in their academics, behavior, and self-esteem. This progress is mainly due to the high expectations AHP administration and team have of the children and the love and stability the children feel because of the long-term nature of the home.

AHP also strives to erase the stigma associated with foster care by utilizing a family-oriented approach in caring for the children, which promotes a sense of belonging and accountability. This approach is a unique style that is engrained in the mission of AHP and a character trait that all AHP staff must possess.

Aunt Hattie's Place has a solid reputation for working with children through their problems and an excellent rapport with the Department of Human Resources and the Department of Social Services.

1.2 Programming Language

The basic programming language I used in developing Easy Math is Microsoft Visual Basic. Mainly because of the mastery I have over the language and the availability of the software.

1.3 Why Easy Math?

Easy Math is a highly customized software package used in facilitating and complementing younger students in learning basic arithmetic. There are tons of software packages in today's market which are geared towards achieving this same goal. However, AHP chose to develop its own package rather than buy a commercial software package. Generally, from a Technology Educational standpoint, educational software should focus on five main functions; **drill and practice** (drill and practice provides exercises for students to solve and receive immediate feedback), **tutorials** (an instructional sequence similar to a classroom lecture), **simulations** (a computerized image of a real or imaginary system used to teach how the system works), **fun and games** (software designed for learning using game rules as a source of motivation),and **problem solving** (provides opportunity to practice solving a particular content area). The different already existing packages should touch one or more of the five software functions stated above. Some of the already existing software for kids include; ***Superkids educational software, Easy Learning software,*** **and** ***Math Media***. These software packages mainly exploit the fun and games, and the drill and practice software functions. Fun and games has been the function of choice for most developers who believe that the best way of teaching younger students is through entertainment. This method has been very successful over the years. However, some educational technology researchers have criticized the usage of fun and games. The main reason for criticism being that some times it is very difficult for the younger students to focus on what really is important. Younger students using fun and games without supervision will most likely focus more on the entertaining factor of the software than on the educational factor.

Another popular software function used by most developers (especially those developing software for learning mathematics/arithmetic) is the drill and practice. It is very common and also very useful.

At AHP, we decided to develop a package that was different from what is commonly being sold in today's market. AHP administrators wanted a cost effective package produced by some one they could trust. They also preferred to put the development in the hands of some one who knows the individuals who would use the software.

We then decided to develop a package that primarily eliminates fun and games to ensure that the user(s) focuses on the things we deemed were most important. Then, we decided to use three of the five main software functions (tutorials, drill and practice, and problem solving). Using these three functions make our package unique as it is very unusual to see a commercial software package that exploits these three software functions. Generally most packages use fun and games, some others exploit drill and practice while others go as far as incorporating both. For Easy Math, I decided to use drill and practice which I think is inevitable for an arithmetic oriented package like this, then I used tutorials. Tutorials should serve as a refresher for the user(s). Finally, I used the problem solving function so that at the end, the user can be properly tested.

We strongly believe Easy Math is at the moment the most appropriate package for the young men at AHP because it uses three main software functions.

Chapter 2

Navigating the Easy Math Software Package

Easy Math was intended specifically for young men who do not necessarily have a solid background in computer usage. With that in mind, I had to ensure that the package was user friendly and navigation was quite easy and straight forward. Figure 2.1 shows the data flow diagram for the user interface.

When the program is launched (i.e. at run time) the Introduction form (a welcome form) is loaded. This form simply prompts the user for a log in name. Once log in name is provided, the user is taken to the main navigation form (user chooses what he/she wants to do next). At this stage, the user has two options; Tutorials, or Tests. On choosing Tutorials, the user is taken to a Tutorial main page where he/she can choose between the six available lessons (addition, subtraction, multiplication, division, signed numbers, and fractions). On the other hand, if the user chooses Tests, then he/she is taken to the main testing page. On the main test page, the user has three main options; drill and practice, practice tests, and final test. The user may log off after completing the final test and receiving feedback. (See Figure 2.1)

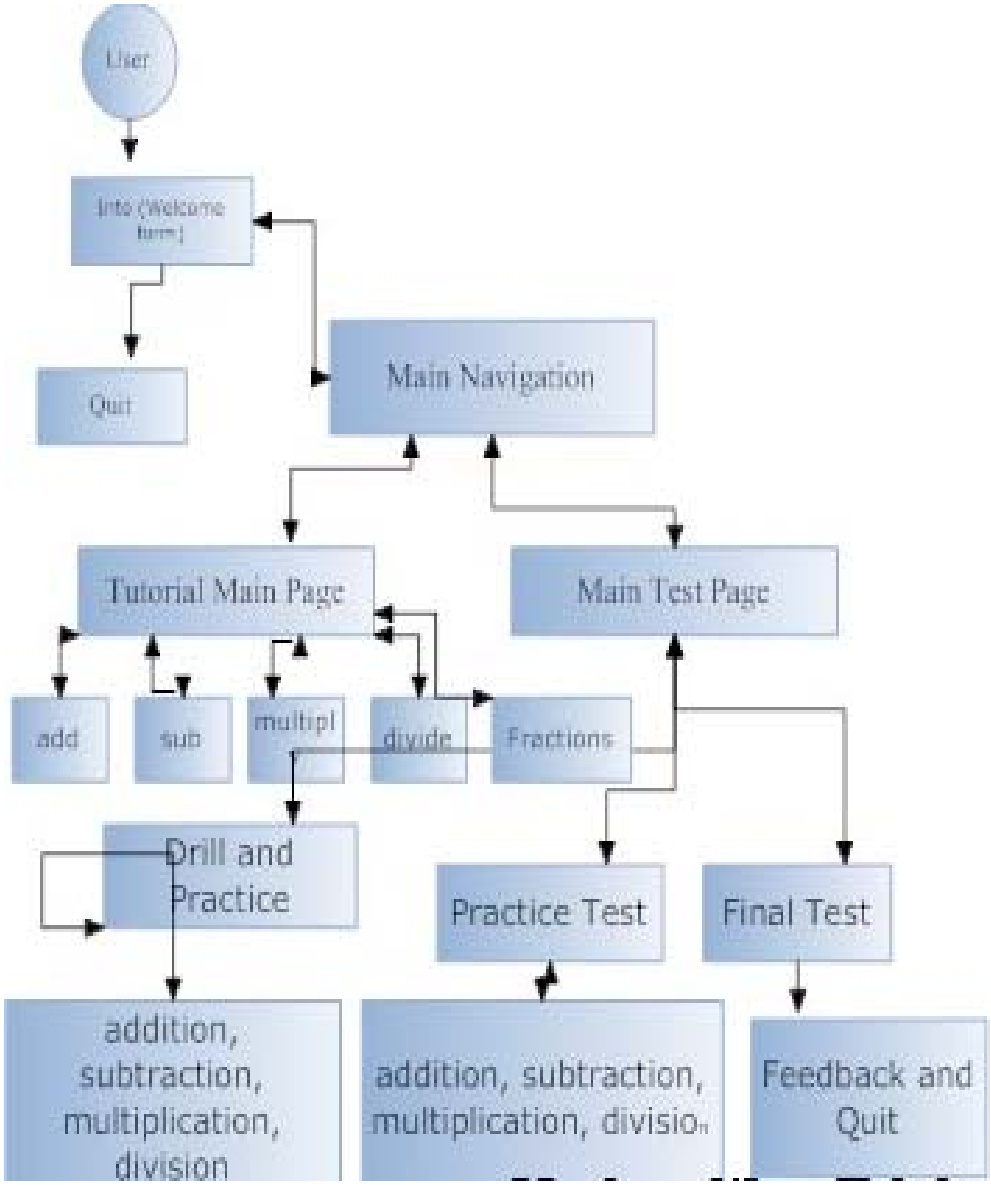

Figure 2.1

User Interface Diagram showing Data Flow

Chapter 3

Easy Math

Easy Math software package was created in approximately six months for AHP and it is made up of 18 forms exploiting the ideas of tutorials and drill and practice. The forms are very user friendly which facilitates navigation of the software package. The forms used include: An Introduction form (a welcome form), Main navigation form, Tutorial main form, Addition tutorial form, Subtraction Tutorials form, Multiplication Tutorial form, Division Tutorial form, Signed Numbers Tutorial form, Fraction Tutorial form. Also, there is a main Test form, and separate practice test forms for addition, subtraction, multiplication, and division. Finally there is a main final test form.

3.1 Project

Introductory Form- The Start Form

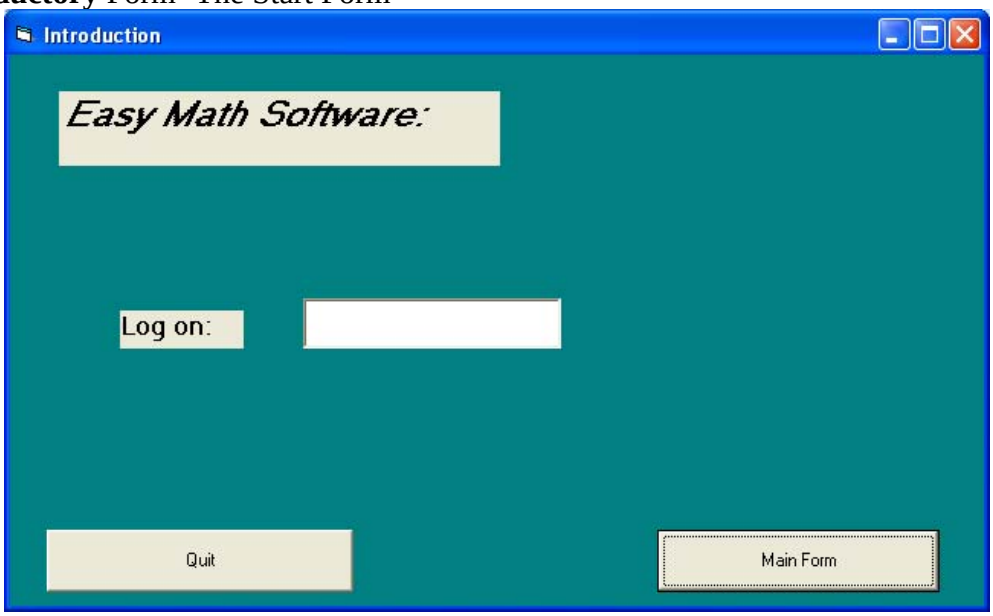

Figure 3.1 Main Entry Form: Introduction form

The introduction form (Figure 3.1) is launched once the program starts running. This form contains two labels, a textbox, and two command buttons. The form is user friendly and permits for very easy navigation. The first label contains the title of the software: "**Easy Math Software**". The next label contains the text "**Log on**" which simply guides the user on where to type their name. An empty text box is provided for the user to type in his/her user name. Then there are two buttons: a "**Quit**" command button and a second command button named "**Main Form**". On clicking the Main form button the user may or may not be redirected to the main navigation form. On clicking the Main form button, (after putting in a username) the software performs a validation process for the user using an **IF ELSE IF statement.** (i.e. checks if the username matches with the authorized user names). If there is a match, the user is redirected to the main navigation form (Figure 3.4).

```
If txtlog.Text = "Betow" Then
        Intro.Hide
        Main.Show
ElseIf txtlog.Text = "Baltimore1" Then
        Intro.Hide
        Main.Show
ElseIf txtlog.Text = "Mont1" Then
        Intro.Hide
        Main.Show
Else
MsgBox "Try Again!!!" vbExclamation, "Incorrect Log"
End If
```

Figure 3.2 If statement to check for validity of log on

If not, the user is prompted to try again using a customized message box. (Figure 3.3)

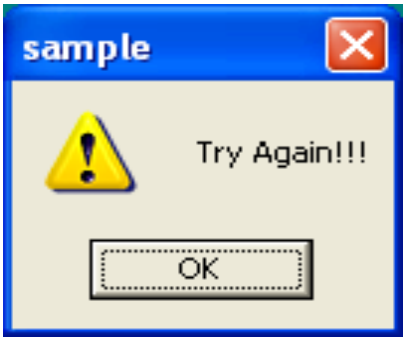

Figure 3.3 Message Box for bag logging

On the other hand, if the user clicks the "**Quit**" command button he/she will be logged off. In summary, from this form the user has the option of moving to the main navigation form or to exit the software and return later.

Main Navigation Form

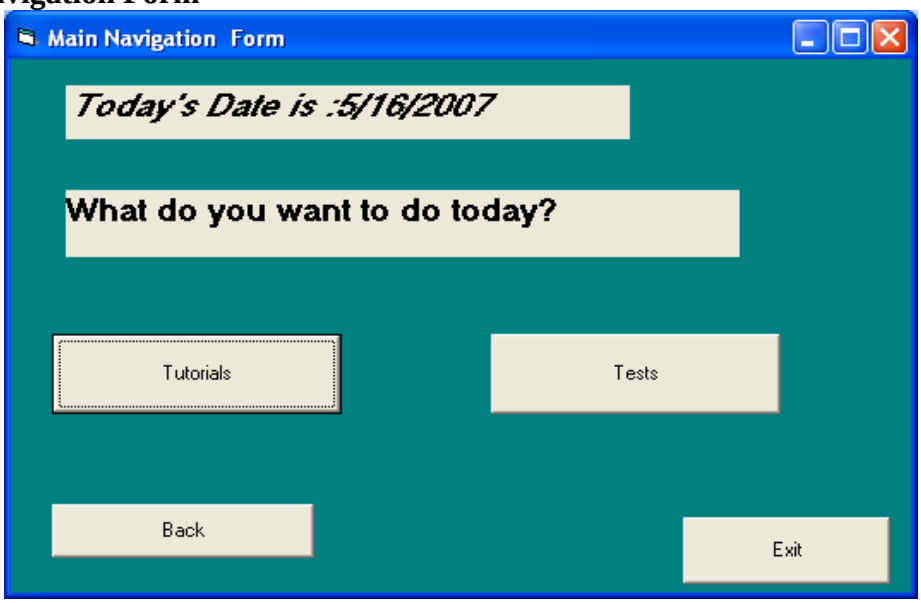

Figure 3.4 Main Navigation Form

The main navigation form gives the user the opportunity to decide on what he/she intends to do in a given session. The form actually consists of two labels, and four command buttons. The first label is the date label which automatically loads and updates the day's date and the time of the session. The second label simply contains the text "**What do you want to do today**?" a simple prompt for the user to choose what to do. There are four command buttons. The buttons are appropriately named such that navigation is easy even for a first time user. The first command button is the tutorial button "***Tutorial***" which allows the user to move onto the tutorial main page. Once the user chooses Tutorials, "***Main Form"*** is hidden (disappears) and only the "***Tutorial"*** form is visible. The second command button with the caption "***Tests***" allows the user to move on to the main testing page. Once this button is clicked, the "***Main Form"*** is hidden and only the "**Tests"** form is visible. The third command button has the caption "***Back***". The ***"Back"*** command permits the user to go back to the previous form. The fourth and last button is captioned ***"Exit".*** The ***"Exit"*** button allows the user to log off the software at that point (without going any further or going backwards). From the main navigation form, the user can either move to the main Tutorial form (figure 3.5) or to the main Test form as shown in Figure 3.5.

Tutorial Form

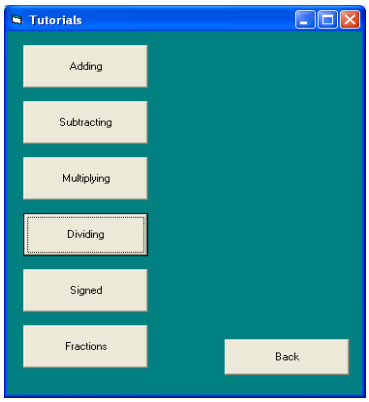

Figure 3.5 Tutorials Main Form

The tutorial form contains seven command buttons. The first command button captioned "***Adding***" redirects the user to a form containing short notes and examples on adding numbers between zero and twenty. The second button with the caption "***Subtracting***" redirects the user to the form containing tutorials on how to subtract numbers between zero and twenty. The "***Multiply***" command button takes the user to the form with notes on how to multiply numbers between one and twenty. The "***Division***" button takes the user to the page with notes on how to divide numbers between one and twenty. The next command button captioned "***Signed Numbers***" permits the user to move to the next form with short notes, pointers, and examples on how to manipulate signed numbers. The "***Fractions***" command button takes the user to the form containing short notes and examples of how to work with fractions. The last command button is captioned "***Back***". This command button allows the user to go back to the previous page (main navigation form). The user is not given the chance to exit the software at this stage. To quit, the user has to go back to the main navigation form before exiting.

Tutorial Lessons

Included in this software package are six short tutorials. There are tutorials on addition, subtraction, multiplication, as well as division of numbers between one and twenty. Also included are short notes and tricks or techniques on how to work with fractions and signed numbers (-1, -6, -8 etc). These tutorials are all on separate forms. The six tutorials were created in Microsoft Word and then later uploaded into the forms using the OLEDB property.

3.2 Tests Form

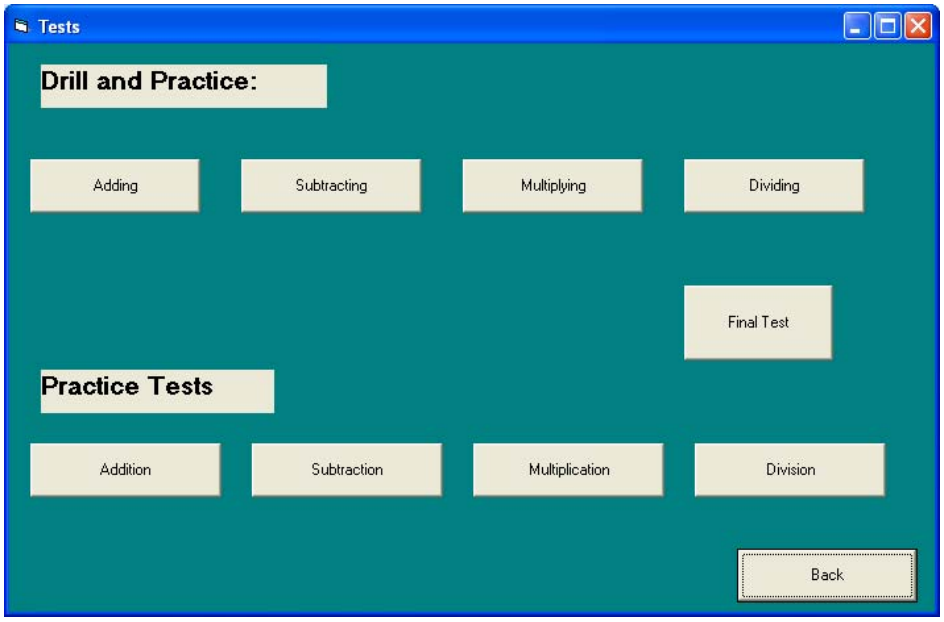

Figure 3.6 Main Test form

The main "***Tests***" form is made up of ten command buttons and two labels. The first label contains the text "***Drill and Practice***" and the second label has the text: "***Practice Tests***". Below the "***Drill and Practice***" label, there are four command buttons captioned "Adding", "Subtracting", "Multiplying" and "Dividing". On clicking on any one of those command buttons, the user is redirected to a drill and practice form for one of addition, subtraction, multiplication, or division.

Below the "***Practice Tests***" label are four other command buttons captioned "Addition", "Subtraction", "Multiplication" and "Addition". Clicking on any of these command buttons takes the user to four separate practice test forms; a test for addition, one for subtraction, one for multiplication and one for division.

The ninth command button captioned "***Final Test"*** takes the user to a single combined testing form. The test consists of all addition, subtraction, multiplication, and division of numbers (signed and unsigned) between one and twenty.

The final command button has the caption "***Back***" which allows the user to go back to the previous form. This also ensures that the user can not exit the software at this stage. To exit the user has to go back to the main navigation form.

Basically from the test form the user can go to any of the four drill and practice forms, or to any of the four practice test forms. The user can also choose to move onto the final test form.

Drill and Practice forms (4 main forms)

There are 4 main forms: the first is called "***Adding Practice***". This provides the user with an opportunity to input 2 numbers to add and see what the result would be.

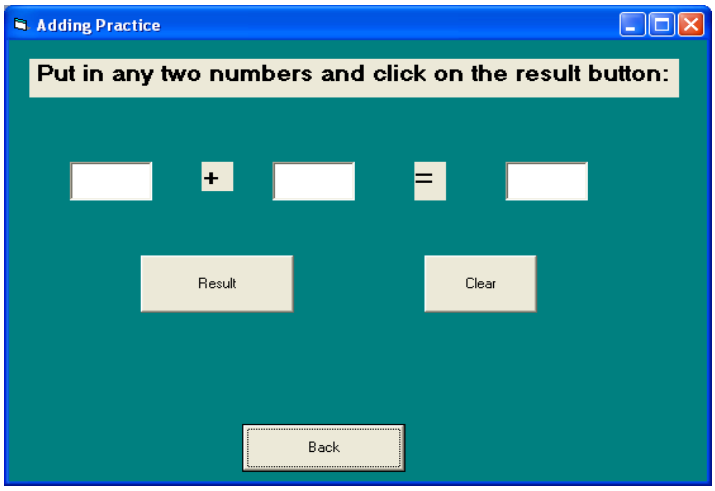

Figure 3.7 Adding Practice Form

This form has three labels, three text boxes and three command buttons. The first two text boxes are the user input boxes. The user inputs any two numbers and then the third text box is the output box (where the user gets the result). So once the user inputs the two

numbers, he/she has to click on the first command button, "***Result***", to obtain the sum. When the user inputs a number into the first text box, that number is stored in a predefined integer variable a. ***(Integer a = txt1.text)***. When the user inputs the next number into the second text box, that number is also stored in a predefined integer variable b ***(Integer b = txt2.text*)**. When the user clicks on ***Result***, these two integers are added up and the sum is made available to the user using the third text box ***(txt3.text = a + b).*** In performing this addition, we have to store the inputs from the textboxes as integers to allow us to add them up.

```
Private Sub Result_Click()
Dim a As Integer
Dim b As Integer
a = txta1.Text
b = txta2.Text
txta3.Text = a + b
End Sub
```

Figure 3.8 Code used for adding two integers

Otherwise, the computer simply concatenates them and thus adding up 11 + 11 will yield 1111 instead of 22. The second command button is captioned "***Clear***" and like its name indicates, it allows the user to clear off the three used text boxes and continue (inputting some more numbers) working.

```
Private Sub Clear_Click ()
txta1.Text = ""
txta2.Text = ""
txta3.Text = ""
txta1.SetFocus
```

Figure 3.9 Code to clear textboxes

Subtracting Practice

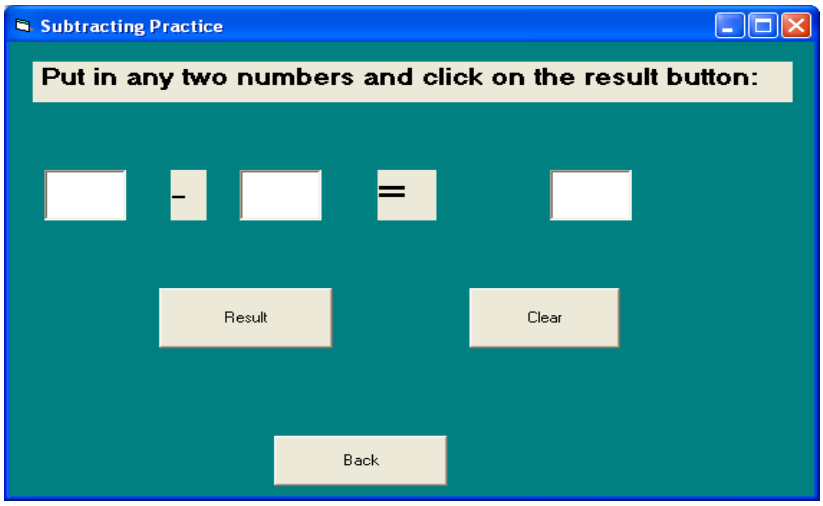

Figure 3.10 Subtracting Practice Form

The second form is called the "***Subtracting Practice***" form. This form is physically the same as the Adding Practice form. It allows the user to put in any two numbers and find their difference. Two text boxes are provided for user input and a third text box for the software output. After inputting any two numbers using the two input text boxes, the user clicks on the ***Result*** command button to obtain the difference. On clicking the result button, what actually happens is the input text from the second text box is subtracted directly from the input text from first text box (***txt3.text = txt1.text – txt2.text***).

```
Private Sub Result_Click ()
txts3.Text = txts1.Text - txts2.Text
End Sub
```

Figure 3.11 Code for subtracting two integers

After which he/she can click on the ***Clear*** command button to clear off the text boxes and continue.

```
Private Sub Clear_Click ()
txta1.Text = ""
txta2.Text = ""
txta3.Text = ""
txta1.SetFocus
```

Figure 3.12 Code to clear up textboxes

Unlike the addition practice form, the inputs from these text boxes don't have to be stored in integer variables before subtraction.

Multiplying Practice form

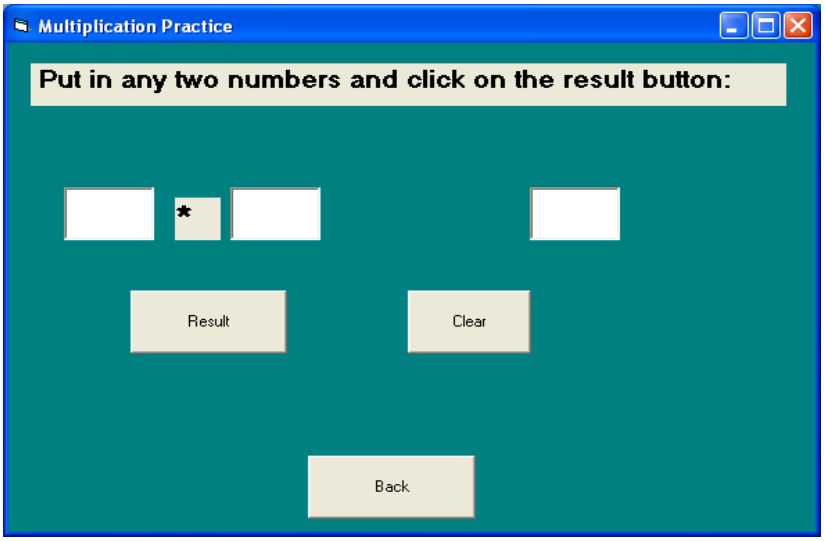

Figure 3.13 Multiplying Practice Form

This form looks physically the same as the Adding and Subtracting practice forms. It allows the user to put in any two numbers and find their product. Two text boxes are provided for user input and a third text box for the software output. After inputting any two numbers using the two input text boxes, the user clicks on the ***Result*** command button to obtain their product. Afterwards he/she can click on the ***Clear*** command button to clear off the text boxes and continue. On clicking the result button, the input text from the first text box is directly multiplied with the input text from second text box and the result is outputted in the third textbox (***txt3.text = txt1.text*** * ***txt2.text***).

```
Private Sub Result_Click ()
Dim a As Integer
Dim b As Integer
```

```
a = txtm1.Text
b = txtm2.Text
txtm3.Text = a * b
End Sub
```

Figure 3.14 Code for multiplying two integers

Dividing Practice form

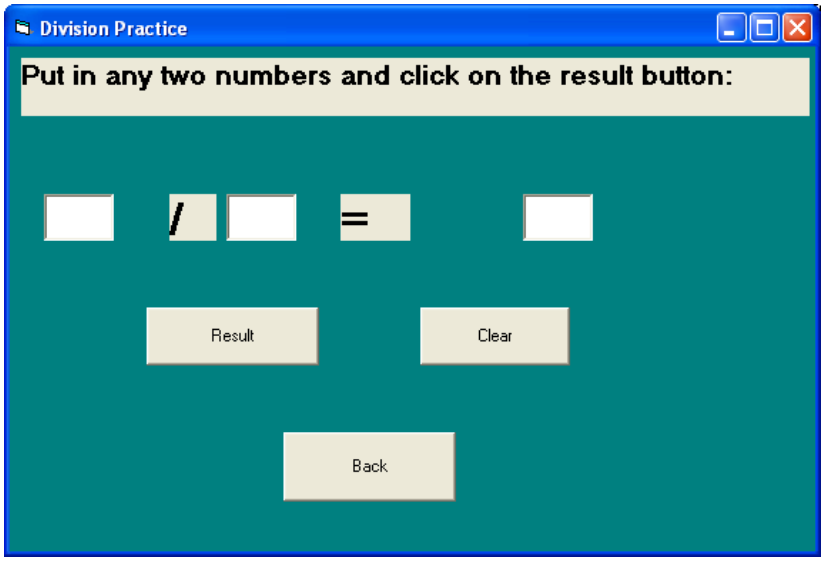

Figure 3.15 Division Practice Form

The division practice form is very similar physically to the three previous practice forms. It allows the user to put in any two numbers and find their quotient. Two text boxes are provided for user input and a third text box for the software output. After inputting any two numbers using the two input text boxes, the user clicks on the ***Result*** command button to obtain their quotient. After which he/she can click on the ***Clear*** command button to clear off the text boxes and continue. On clicking the result button, the input text from the first text box is directly divided by the input text from second text box and the result is outputted in the third textbox (***txt3.text = txt1.text*** * ***txt2.text***). Like with the subtracting and multiplying practice we directly divide the input obtained from the textboxes with out actually storing them in integer variable.

```
Private Sub Result_Click ()
txtd3.Text = txtd1.Text / txtd2.Text
End Sub
```

Figure 3.16 Code for multiplying two integers

Practice Test Forms

There are four main practice test forms: ***Addition, Subtraction, Multiplication***, and ***Division***. All the forms look the same physically. However, there is a little difference in the coding for the addition and the division.

Addition form

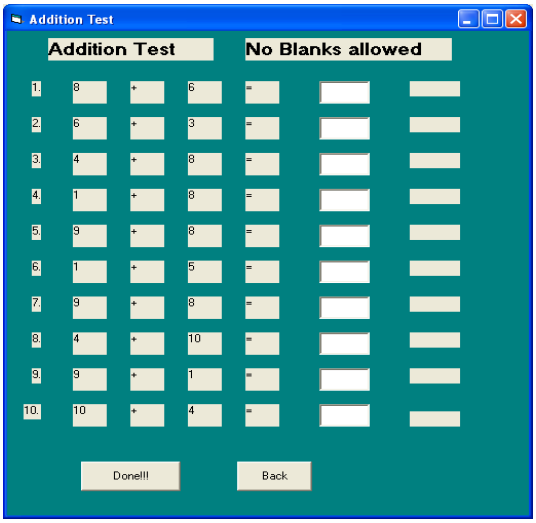

Figure 3.17 Addition Practice Test Form

This form permits the user to take a practice test in addition. There are 10 randomly generated addition questions and 10 textboxes for the user to input his/her answer.

Random Number function:

```
Dim low As Integer
Dim high As Integer
Public Function Rand(low, high) As Integer
 Rand = Int((high - low + 1) * Rnd) + low
End Function
```

Figure 3.18 Random number function

The questions are generated and outputted using labels (***label1a.caption = rand (low, high***)). Every time the user loads this form, a new test is generated. When the user answers all 10 questions, he/she clicks on the "***Done***" command button. On clicking the "***Done***" command button, the test is checked and the text "**ok**" is printed next to each correctly done question using a label. The numbers generated randomly are stored in pre defined integer variables (Integer a = label1a.caption). When the user selects the done command button, an "***ok***" is printed next to the correct ones and a "***No***" is printed next to the incorrectly done questions (if txtans1.text = a + b, then lalbel1d.caption = "ok" else lable1d.caption = "No").

Subtraction

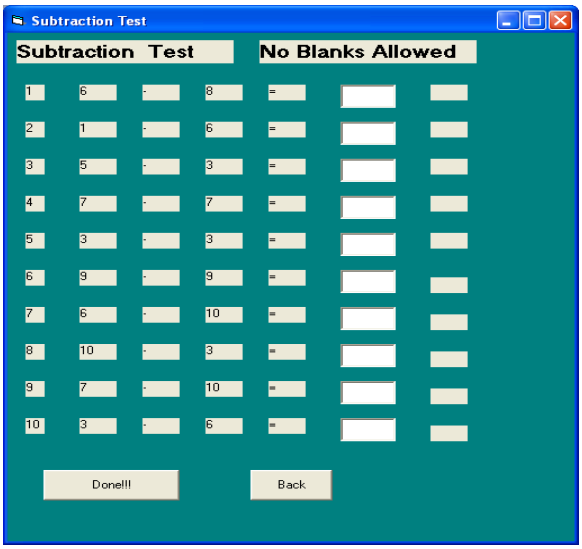

Figure 3.19 Subtraction Practice Test Form

This form permits the user to take the practice test in subtraction. There are 10 randomly generated subtraction questions and 10 textboxes for the user to input his/her answer. The questions are generated and outputted using labels (label1a.caption = rand (low, high)).

Every time the user loads this form, a new test is generated. When the user answers all 10 questions, he/she clicks on the "***Done***" command button. On clicking the "***Done***" command button, the test is checked and the text "**ok**" is printed next to each correctly done question using a label. Unlike in the addition practice test, the randomly generated test numbers are not stored in predefined integers. The addition is done directly using the labels. When the user selects the done command button, an "***ok***" is printed next to the correct ones and a "***No***" is printed next to the incorrectly done questions (if txtans.text = label1a.caption + label1b.caption, then lalbel1d.caption = "***ok***" else lable1d.caption = "***No***"). After taking the practice test the user has to click on the "***Back***" button to return to the main test page.

Multiplication

Figure 3.20 Multiplication Practice Test Form

This form permits the user to take the multiplication practice test. There are 10 randomly generated multiplication questions and 10 textboxes for the user to input his/her answer. The questions are generated and outputted using labels (label1a.caption = rand (low,

high)). Every time the user loads this form, a new test is generated. When the user answers all 10 questions, he/she clicks on the "***Done***" command button. On clicking the "***Done***" command button, the test is checked and the text "**ok**" is printed next to each correctly done question using a label. Unlike the addition practice test, the randomly generated test numbers are not stored in predefined integers. The addition is done directly using the labels. When the user selects the "***Done***" command button at the end of the test, an "***ok***" is printed next to the correctly done ones and a "***No***" is printed next to those that are incorrect (if txtans.text = label1a.caption * label1b.caption, then lalbel1d.caption = "***ok***" else lable1d.caption = "***No***"). After taking the practice test the user has to click on the "***Back***" button to return to the main test page.

Division

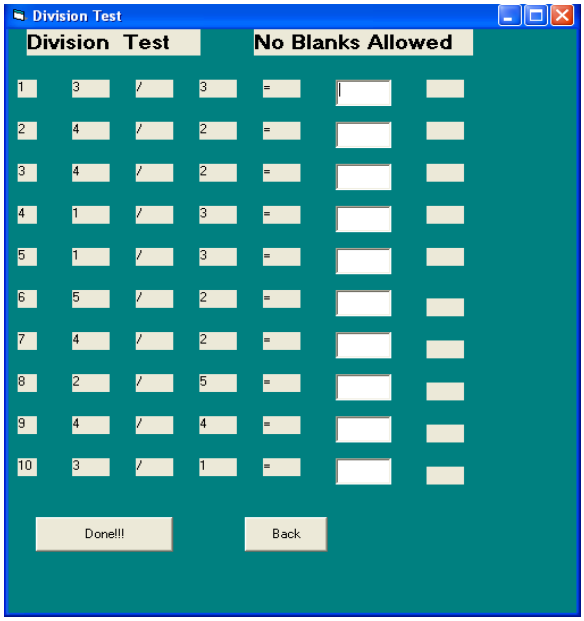

Figure 3.21: Division Practice Test Form

This form permits the user to take the division practice test. There are 10 randomly generated division questions and 10 textboxes for the user to input his/her answer. Unlike the previous forms, the random numbers are generated strictly from 1 to 50 (high = 50

and low = 1) to avoid division by 0. The questions are generated and outputted using labels (label1a.caption = rand (low, high)). Every time the user loads this form, a new test is generated. When the user answers all 10 questions, he/she clicks on the "***Done***" command button. On clicking the "***Done***" command button, the test is checked and the text "**ok**" is printed next to each correctly done question using a label. Unlike in the addition practice test, the randomly generated test numbers are not stored in predefined integers. The addition is done directly using the labels. When the user selects the "***Done***" command button at the end of the test, an "***ok***" is printed next to the correctly completed ones and a "***No***" is printed next to those that are incorrect (if txtans.text = label1a.caption / label1b.caption, then lalbel1d.caption = "***ok***" else lable1d.caption = "***No***"). After taking the practice test the user has to click on the "***Back***" button to return to the main test page.

Final Test Form

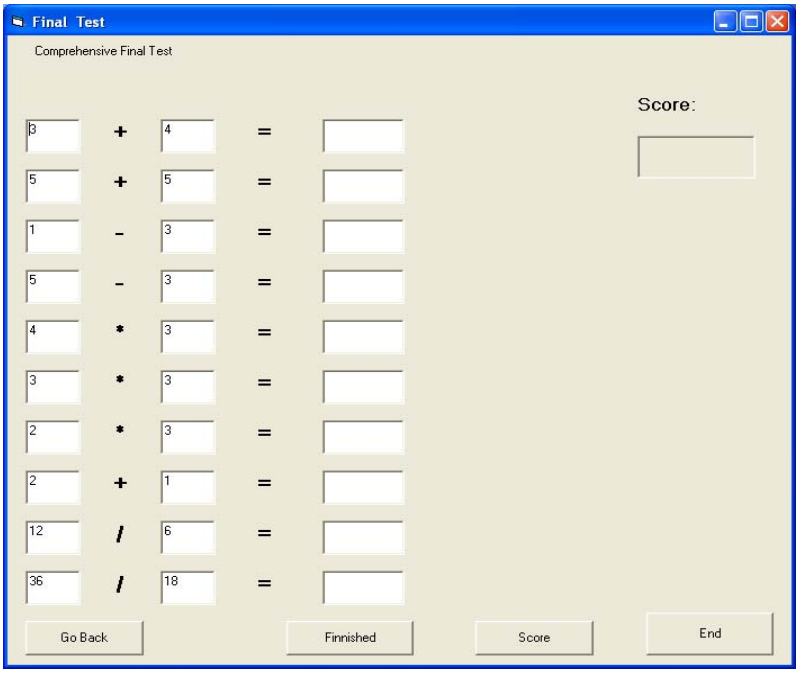

Figure 3.22: The Final Test Form

The final test form serves mainly for evaluation of what the user has gained from using the tutorials, the practice tests, and the drill and practice examples. This form is made up of several textboxes with questions randomly generated and more textboxes for the user to input his/her answer. At the end of the test, the user clicks on the "***Finished***" button. On clicking the finished button, the user can tell the correctly answered questions from the incorrect ones.

There will be an "***ok***" next to the correct ones and a "***No***" next to the incorrect ones. The user can then click on "**Score**" to get his/her score in percentage. The scores are outputted with the use of a picture box.

```
Private Sub Score_Click ()
Dim k As Integer
Dim z  As Integer
Dim Score As Integer
Score = 0
For k = 0 To 9
If lblans(k).Caption = "ok" Then
Score = Score + 1
End If
Next k
z = Score * 10
scorebox.Print z; "Percent"
If Score > 6 Then
MsgBox "Good Job   !!!" vbExclamation, "sample"
Else
MsgBox "Try Again...  !!!" vbExclamation, "sample"
End If
End Sub
```

Figure 3.23 Code for evaluation of the final test

After checking his/her score the user may return to the previous screen (Test Form) or may exit by clicking on the "***End***" command button.

Chapter 4

Difficulties in Coding

The principal difficulties I encountered while working on this project were in two folds.

- Understanding the exact requirements of the customer
- Coding of the tests and evaluations

4.1 Understanding Customer Requirement

This stage of the product was slightly difficult because the customer was not very sure of what exactly she wanted. I worked on a sample for a few weeks and asked for her input. This was really helpful as it helped broaden her view as to what kinds of features we could possibly add.

Coding of the Test and Evaluations

- **Using an array of text boxes and labels**

Although this task was not very difficult, however, it was my first time using arrays for text boxes and labels. The main issue was ensuring that the textboxes and labels were lined up chronologically. If not properly handled, it can lead to many mistakes at coding and run time.

- **Adding using the plus sign "+"**

While working on the different addition tests (drill and practice, practice test, and Final test) I could not get the correct answer. In the testing, the numbers are randomly generated into textboxes where they are stored and later on added together (txt1.text = Rand (low, high) and txt2.text = Rand (low, high)). The addition is done by adding the

contents of both textboxes and stored in a third textbox (txt3.text = txt1.text + txt2.text). Using this code, adding 11 and 12 gave me 1112 which implied that the plus sign (+) was not being recognized. Instead, since the randomly generated numbers were stored in textboxes, they were being treated like text and not as integers. So instead of adding the numbers, they were being concatenated (hence: 11 + 12 = 1112). To manage this problem I first of all declared several variables as integers (dim a, b, c, d, e, f as integers) used in storing the numbers in the text boxes (a = txt1.text, b = txt2.text). Finally in the adding stage, I would add up a and b and store in the third textbox (txt3.text = a + b). This solved the addition issue completely.

4.2 Division

I faced a couple of problems with the division tests:

- *Dividing by zero*

The first problem I had while testing the division test was that of dividing by zero. Using the random number generator, every time the number in the second text box was zero, the program would crash since division by zero is not allowed. To manage this issue, I changed the Random function around to ensure that the lowest number that could possibly be generated was 1 and not 0 (Random Function, low = 1 and high = 20). Modifying the random function completely eliminated the issue of dividing by zero.

- *Ensuring that the first number randomly generated is divisible by the second number*

The second problem was that of ensuring that the randomly generated numbers were divisible to avoid having answers with numerous decimal points. Whenever two numbers (like txt1.text = 10 and txt2.text = 3) were randomly generated the result would include

so many decimal places which meant indirectly that the user almost always gets the wrong answer because the computer differentiates 3.3 from 3.3333333. I tried using integer division and that made the situation worse because the computer simply truncates the decimal part (10/3 = 3). To manage this problem I had to ensure that the numbers generated were always divisible. This I achieved by once more modifying the random function. I ensured that the number generated into the first textbox is always a multiple of the number generated in the second text box.

4.3 Correcting the tests, counting the correct answers and outputting the scores using for loops

In coding "Final Test" I used an array of textboxes and labels hoping that it was going to facilitate the general coding. However, I ran into major problems when I attempted to use for loops to check the answers because the questions were a mixture of addition, subtraction, multiplication and division. My intention was to use one ***for loop*** that loops through the entire test checking if the answers in the answer textboxes were correct:

```
For int i = 1 to 10
        If txt (i).text = txt (i).text + txt (i).text then
                Labe1li.caption = "ok"
        Else
                Label1i.caption = "No"
        End if
Next i
```

This loop was meant to go through the entire test and put an "ok" next to the correct answers and a "no" next to the wrong answers. However, I could not use this ***for loop*** because all the questions were not addition questions. To solve this problem, I used ten separate ***if else*** statements to correct the tests:

```
        If txt3.text = txt1.text + txt2.text then
                Label1.caption = "ok"
        Else
```

Label1.caption = "No"

End if

After using these "***If statements***", I used a ***for loop*** to count the number of "***ok***" labels and use that in generating the final score in percentage.

Chapter 5

Hardware and Software Requirements

5.1 Hardware Requirements

There is no real demand on the desktops used in running this Easy Math software program. A basic requirement could be using desktops that have:

- Pentium 1 or higher CPU
- A well functioning mouse and Keyboard
- 256 MB of RAM

5.2 *Software Requirements*

This package was designed using Microsoft Visual Studio 2000 edition. The basic requirement is for the desktops to have Microsoft Visual Basic functional. However, the executable document will run with or without the Visual Studio software.

Chapter 6

Summary and Future Work

6.1 Summary

Easy Math is intended to help the young men at Aunt Hattie's Place improve their already acquired basic arithmetic skills. This application can only complement what they should already know from school. It is in no way intended to replace the teacher. Easy Math as an Instructional software exploits two main functions:

1. Tutorials

Tutorials provide an entire instructional sequence very similar to a teacher's class room notes. These tutorials should help the user to better understand what he/she learnt before. The short notes and tips provided are self contained and are well arranged structure wise so the user does not get mixed up.

2. Assessment (Drill and Practice)

The package also provides drill and practice software functions for assessing the users' progress. Drill and practice software provides exercises in which a user completes some examples and gets a feedback on his/her correctness.

Easy Math package provides a friendly user interface and thus facilitates its general use as a whole. The user does not have to be an expert in computers. The user has to be:

- Capable of reading English
- Capable of using a mouse and
- Capable of using the key board

6.2 Future Work

Easy Math is being used successfully now at Aunt Hattie's Place in Baltimore. However, when I think of future work, one main issue stands out; that of making up a test for the addition, subtraction, multiplication, and the division of fractions. Thus far it has been impossible for me to randomly generate fractions to make up a test.

Also, in the future, I would like to be able to extend the use of Easy math. At the moment I have focused my work on basic arithmetic. However, in the future I would like to extend this to more advanced topics in mathematics like working with radicals and complex fractions.

References:

[1] Gary Bronson, *Introduction to Programming using Visual Basic*. Scott/Jones Inc. 1999

[2]Peter Wright, *Beginning Visual Basic 6.0*. Wrox Press Ltd. August 1998, Source code and support available at: <http://www.wrox.com>

[3] MD Roblyer, *Integrating Education Technology into Teaching*. Merril Prentice Hall' available at: <http://www.prenhall.com/roblyer>

[4] Addition Tutorial 2006. Available at:

<www.intellitools.com/pdf/**tutorials**/classroom_suite/**Addition**%20Tutorial.pdf>

[5] Basic Subtraction 2005. Available at :<hookedonphonics.com>

[6] Multiplication Tutorials 2006. Available at: http://www.schoolcentral.com/

[7] Welcome to Mrs. Lowe's Class. 2006 A student-centered inquiry based learning community. < http://schoolweb.missouri.edu/poplarbluff.k12.mo.us/lowe/math.html>

Digitally signed by John H. Hagen
DN: cn=John H. Hagen, o=West Virginia University Libraries, ou=Acquisitions Department, email=John.Hagen@mail.wvu.edu, c=US
Reason: I am approving this document
Date: 2008.05.10 14:49:21 -04'00'