ClassPad 101

for ClassPad Version 3.00+

Lesson 16
Continuation of Programming

Welcome
In this final lesson on programming the ClassPad we will learn another print command, how to create loops and also how to find programs and MCS data in general in the Exchange Window. By the end of this lesson, you will have enough knowledge to continue the exploration of programming on your own.

Lesson Goals
- To understand the active image
- To understand how to use a For/Next Loop
- To find the distance between two points
- To understand what a program parameter is

In Lesson 16, you will learn how to:
- Create a new vcp file
- Write a program to display formulas in 2-D form
- Use the For/Next loop
- Find the midpoint between a midpoint and endpoint of a segment
- Use parameters in a program
- Run programs from within Main and eActivity

Upon completion of this lesson, you will be able to answer the following questions:
1. How do you step backwards in a For Next loop?
2. Describe two ways to identify which vcp file is active.
3. What command do you use to display math in a natural format?
4. Without running this program, what will be the program's output?
   Hint: Use pencil and paper.

```
ClrText
0→total
For 4→a To 20 Step 4
total+a→total
Next
Print total
```

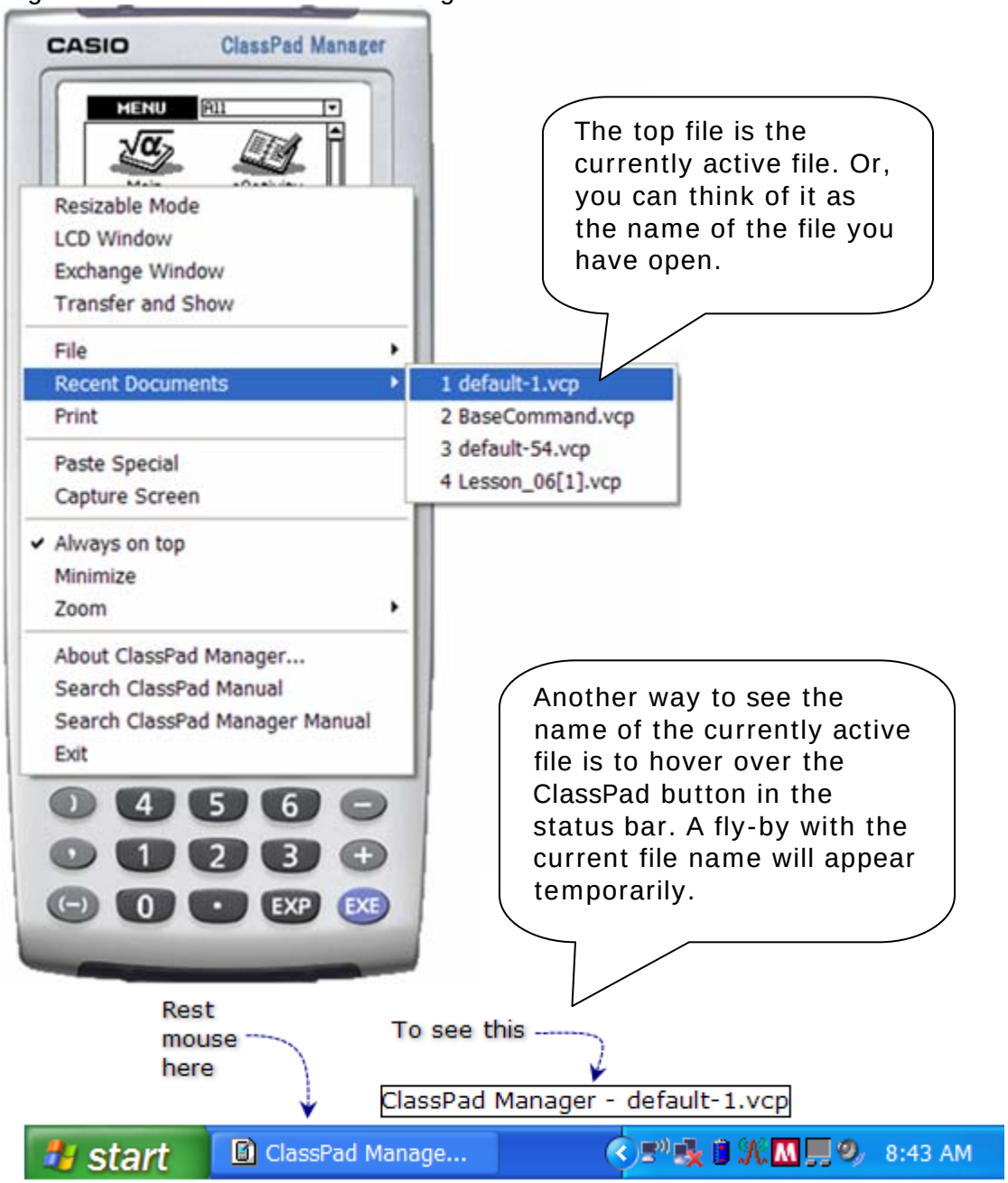
Time required
About 60 minutes.

Getting Started

Where are the programs we created in Lesson 15? They are in the main folder and the main folder is in the MCS memory area. MCS data is stored in the Other Data folder inside a vcp file.

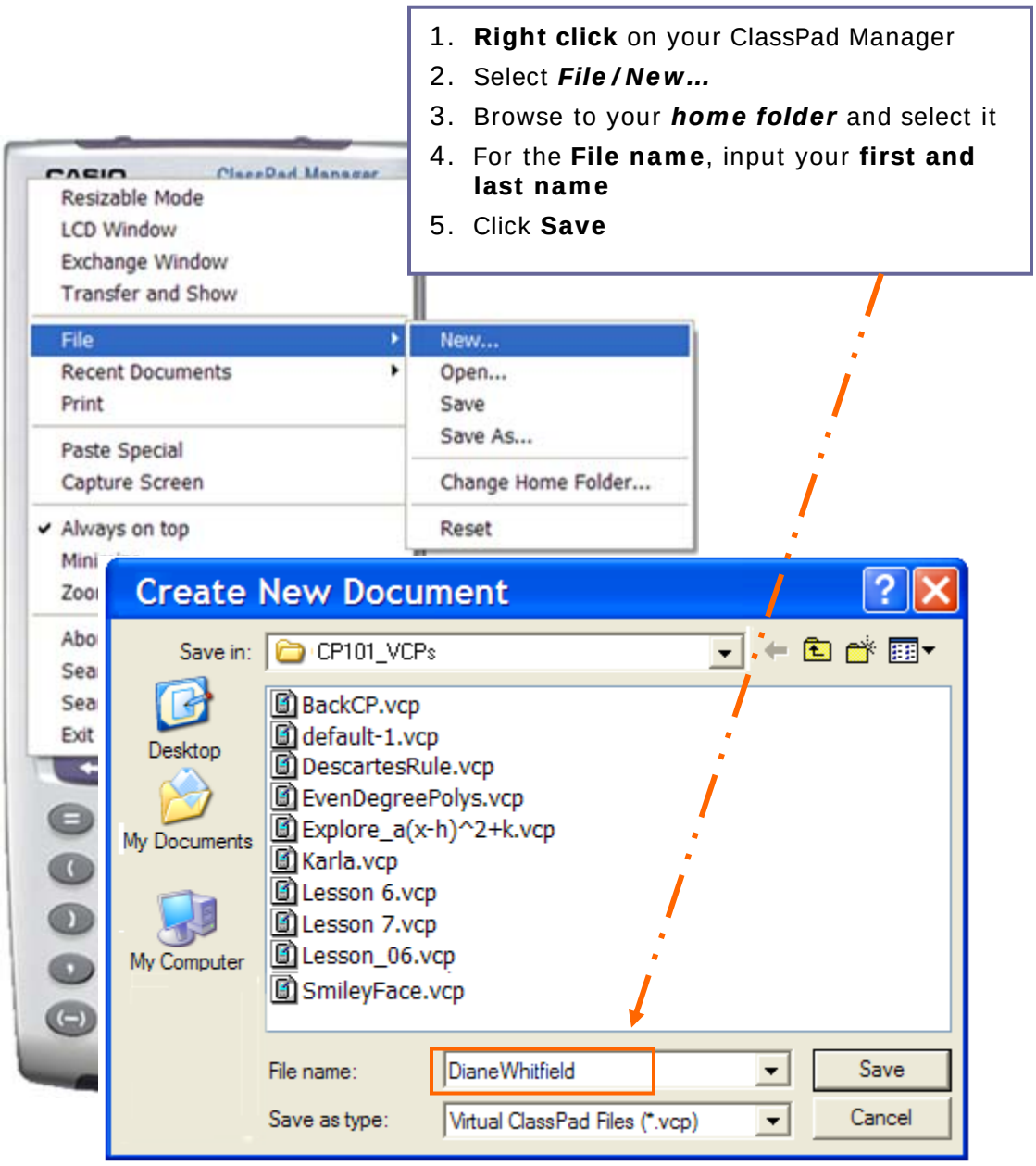
Before beginning the programming parts of this lesson, we will create a new vcp file. But creating a new vcp file, we need the name of our current vcp file.

(A) To find the name of your current vcp file:
Right click on the ClassPad Manager and select ***Recent Documents***:



(B) To create a new vcp file:

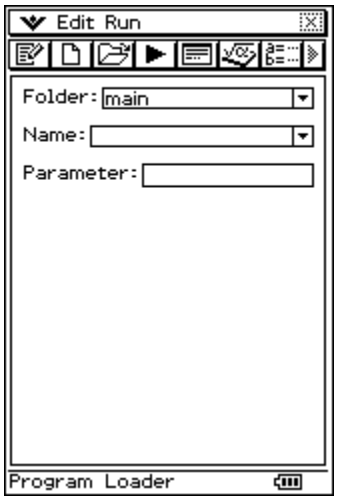
Please create a new vcp file using your first and last name.

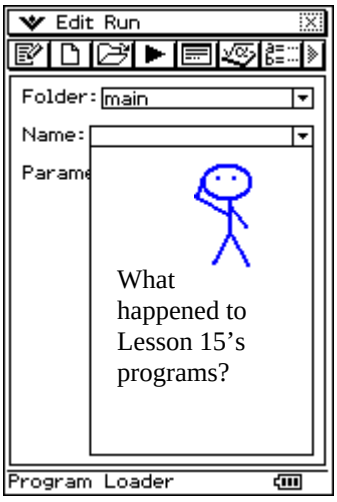


(C) What did we just do?

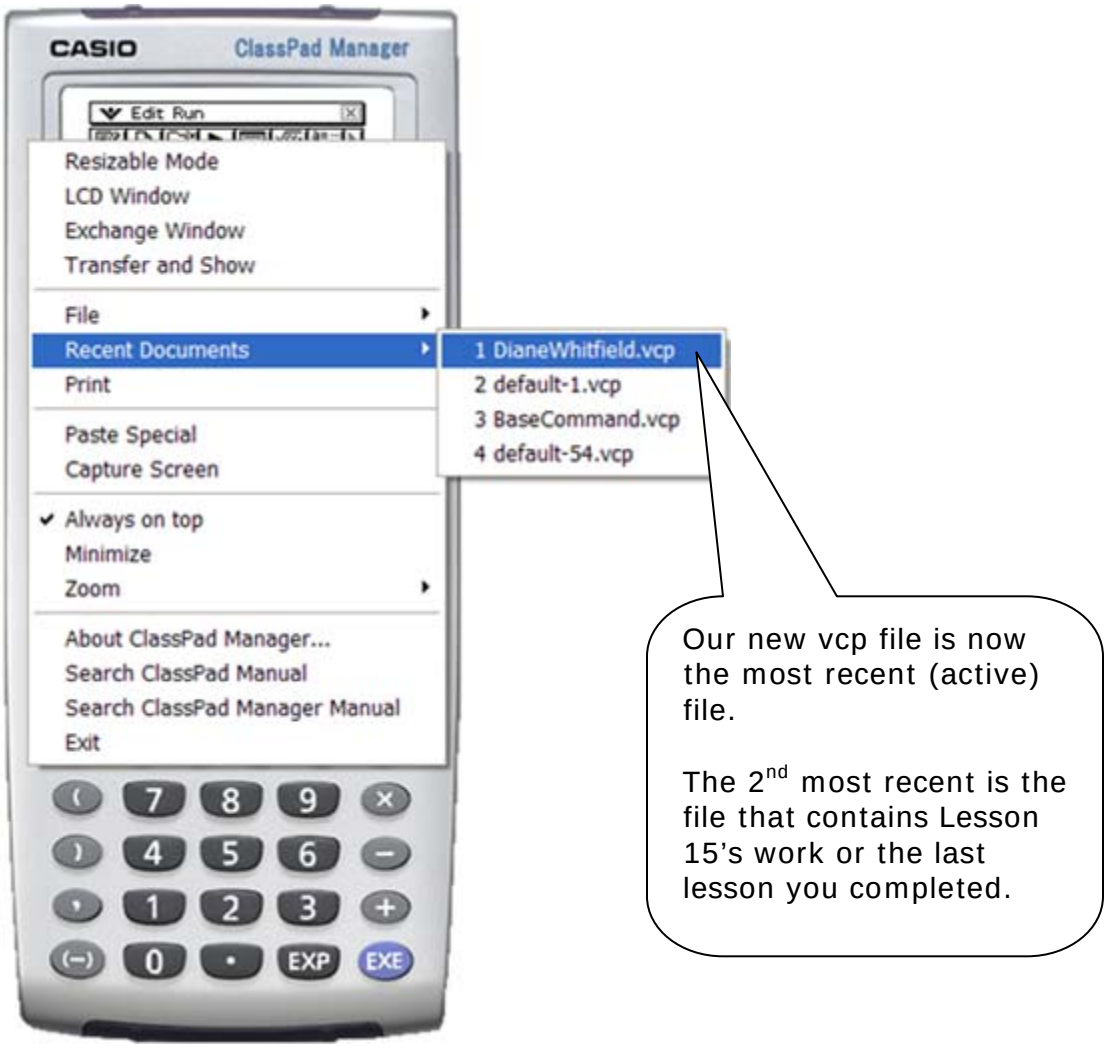
Please open the **Program application** and notice it is empty!

It is empty because you created a new vcp file.





(D) Open the Recent Document list again:



PART I

We will begin this lesson by trying to figure out the *For Next* loop! We have already looked at the If Then control statement. The For Next control statement is another way to control the flow of our program.

This is the general idea of a For Next loop without using special syntax:

When we run this simple program:

```
For i=1 to 10
   Print "In the Loop"
Next i
```

1st i is set equal to 1 and *In the Loop* is printed
Next, i is set equal to 2 and *In the Loop* is printed

Next, i is set equal to 3 and *In the Loop* is printed
.
.
.
Next, i is set equal to 10 and *In the Loop* is printed

Next, i would be 11, but 11>10 and so the program ends ☹

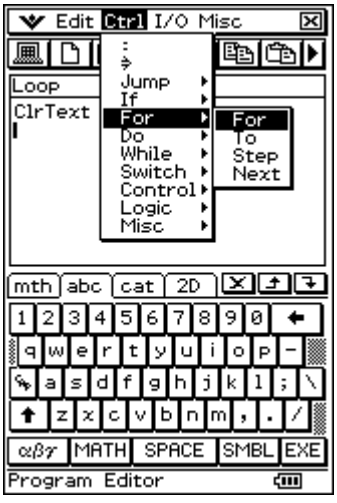
We can also add a *Step* size. For example,

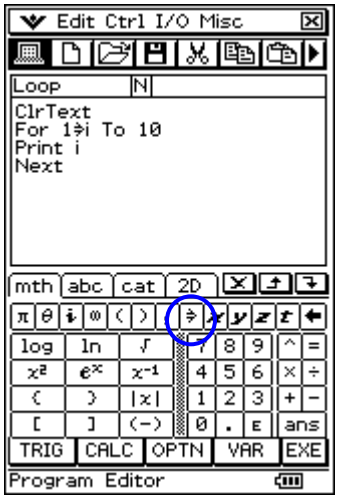
```
For i=1 to 10 Step 3
   Print "In the Loop"
Next i
```

In this mini program, i is first set equal to 1.
Next i steps up to 4 and so on…

1. The ClassPad's Syntax and the For Next Loop

a. Click m and then p
b. Click the O button
c. Type in the name: **Loop**
d. Click **OK**
e. **Type in the program** exactly as it is shown **USING** the commands in the **Ctrl/For submenu** when possible (helps with syntax errors!)
f. **Run your program**…
   The numbers 1 to 10 should appear in the output window





2. Using Step in a For Next Loop and a Counter

Counters are often used in programming. To make a counter, we initialize it at the beginning of the program and then add one to it each time it is encountered.

a. Click in the upper window

b. Click P to open your Loop program

c. Add **Step 3** at the end of the For To statement

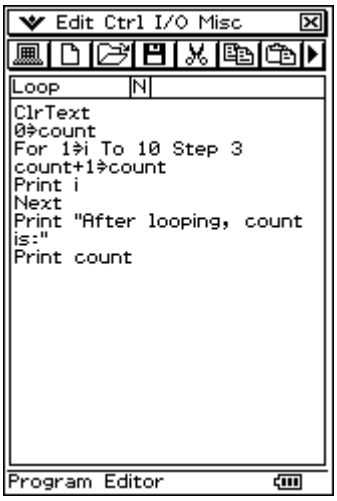
d. **Save** and **run** your program. Did you step by 3 from 1 to 10?

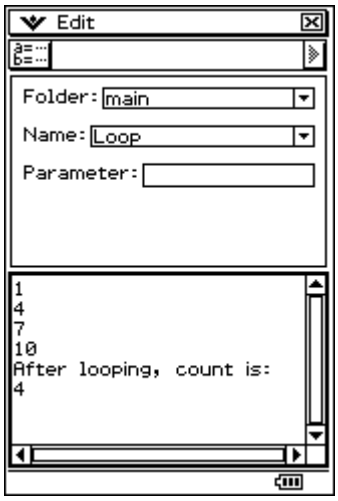
e. **Initialize** a counter by storing **0** in **count**

f. Increment **count** by **adding 1** to it **each time you loop**

g. Print a message following the For/Next loop to display count's value

h. **Run** your program again





3. Stepping Backwards

We can also step backwards by changing the step value to a negative.

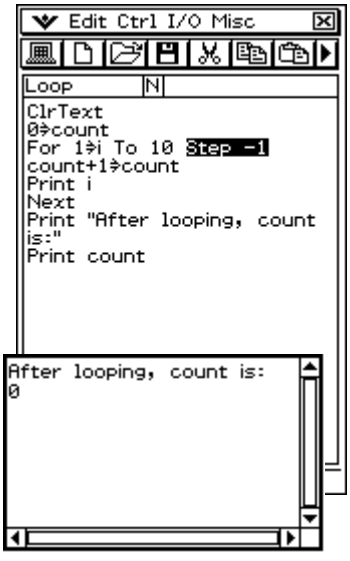
a. In your Loop program change **Step** size to **-1**

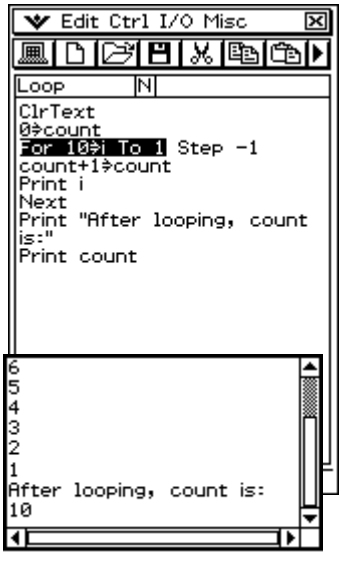
b. **Save and run** your program

c. Well, not what I expected! But, if we start at 1 and step back 1 then we are at 0 and the For loop ends.

d. Change the For loop so that we **start at 10** and **end at 1**

e. **Save and run** your program again

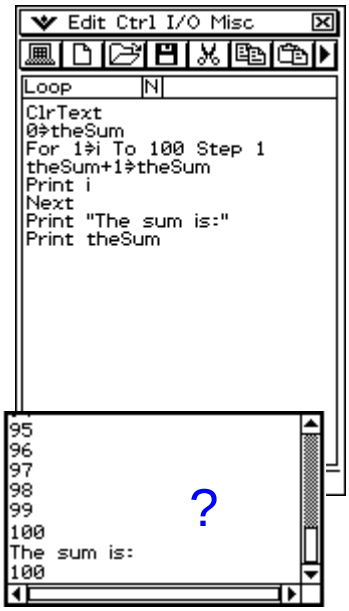




4. Summing Numbers

Let's try summing the numbers from 1 to 100 using the Loop program.

a. **Change** our program to loop from **1 to 100** with a **step of 1**
b. Change each count to **theSum**
c. Change the ending print statements
d. **Save and run** your program
e. Oops, the sum is incorrect…
f. We want to add 1 then 2 then 3 … to theSum, not 1 each time
g. Change theSum+1 to theSum+i
h. Run your program again



Can you check this sum? Think about the numbers…

1, 2, 3, 4, … 97, 98, 99, 100

What pattern did Carl Gauss notice?

2+99=101

1+100=101

A historical note:

Johann Friedrich Carl Gauss was an outstanding German mathematician who contributed many great discoveries to mathematics. He lived from April 30, 1777 to February 23, 1855.

When Carl Gauss was 7 years old, his teacher asked the class to sum the integers from 1 to 100, as a way to keep the students busy. He quickly noticed a pattern, wrote 5050 on his chalk board in less than one minute and turned it in. His answer was correct!

Button Review

- Click  to create a new program.
- Click  to return to the program editor and edit a window.
- Click  to save changes and exit the program editor.
- Click  to run a program.

PART I
Practice Exercises

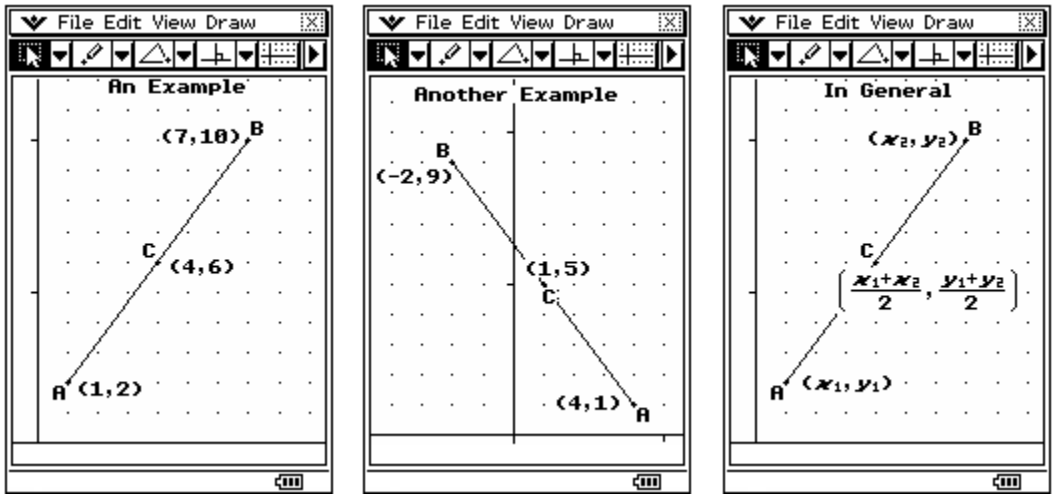
Before beginning the practice exercises, open a word document, type in the following information and then *save it as Lesson16 in your CASIO folder within My Documents*:

- Date: (enter today's date)
- To: (put your instructor's name here)
- From: (put your name here)
- Re: Lesson 16

1. In this practice set, we will modify the Loop program a few times.
2. Change the Loop program to find the sum of the even numbers between 1 and 10.
3. Run your program. With your program output showing, get a **screen capture**. Paste it into your Lesson16 document (under a title of PART I).
4. Change the Loop program to find the sum of the even numbers between 1 and 1000.
5. Run your program to make sure it works.
6. With your program output showing, get a **screen capture**. Add two blank spaces following the first screen capture and then paste this one.
7. Change the Loop program to find the sum of the odd numbers between 1 and 1000.
8. Run your program to make sure it works. [Hint: 1st try running it from 1 to 5; the sum should be 1+3+5 or 9.]
9. With your program output showing the sum of the odd numbers from 1 to 1000, get a **screen capture**. Add two blank spaces following the second screen capture and then paste this one.

PART II

How can we find the midpoint of a line segment? Well, it is not too difficult if we know the end points of the segment. This is how we do it:



1. Displaying the Midpoint Coordinates using PrintNatural

a. Click the 0 button

b. Type in the name: **Midpoint**

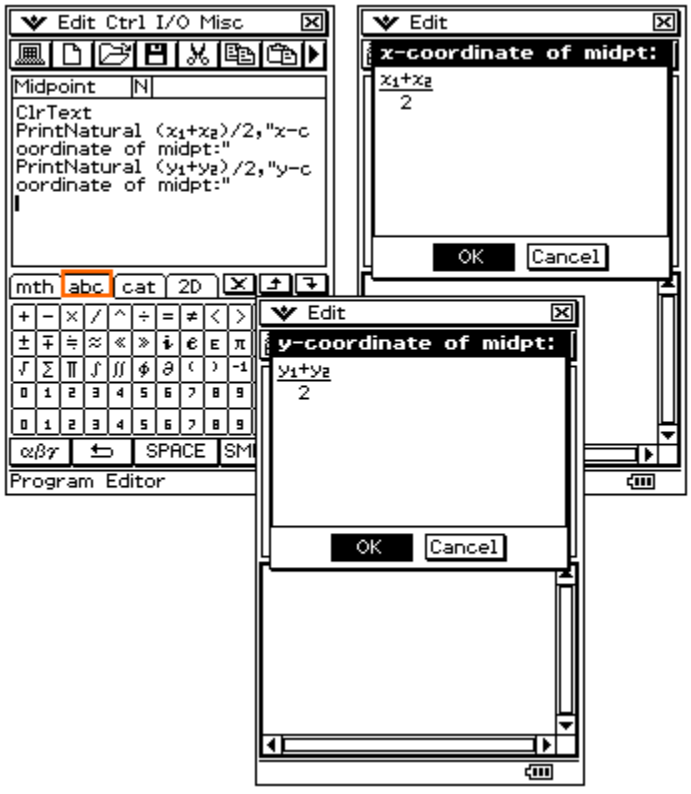
c. Click **OK**

d. Type in the program exactly as shown

e. Note: **PrintNatural** is in the **I/O** menu's **Output** submenu

f. Note: **Subscripts** are in the soft keyboard. Look in the **MATH** button of the **abc** page

g. Run the program and click **OK** to each dialog



2. Getting Input from the User

We need to ask the user to input the endpoints of their line segment as ordered pairs (x, y).

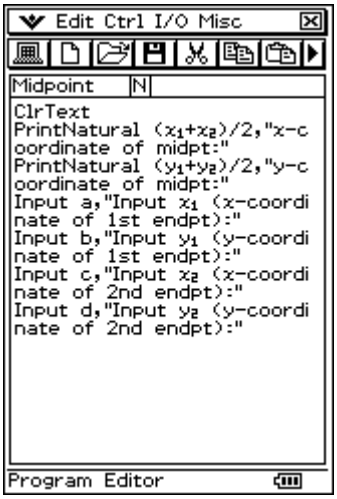
a. Add the following line to your program:

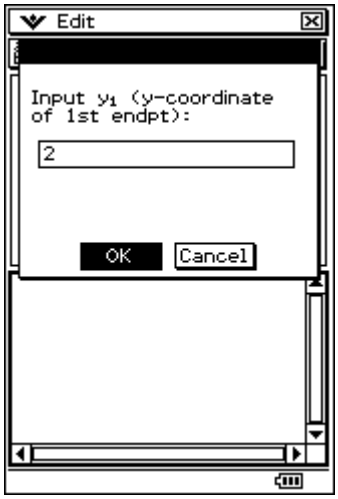
Input a,"Input x_1 (x-coordinate of 1st endpt):"

b. Once you have it typed in, copy, paste and then edit to create inputs for b,c and d

c. Run your program to make sure it runs!

Note: We are asking the user to input **(a, b)** for (x_1,y_1) and **(c, d)** for (x_2,y_2)



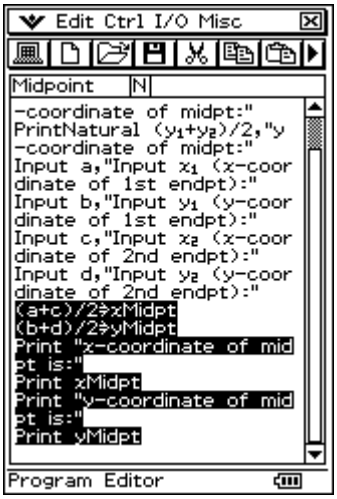


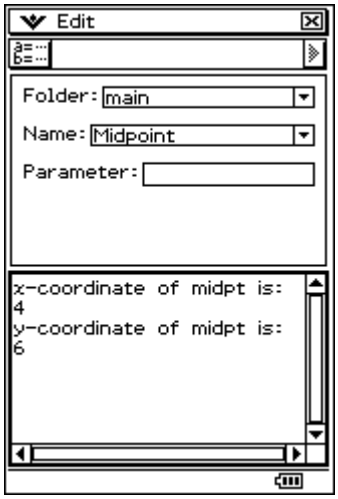
3. Calculating and Displaying the Midpoint

a. **Add six more lines** to your program (think about what each line does as you enter it)

b. **Run your program** using the endpoints from the example at the beginning of the section [(1,2) and (7,10)]

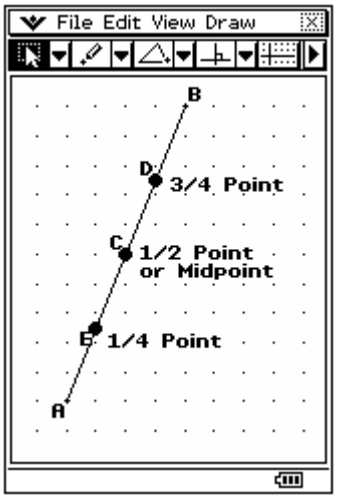
c. Run your program again using the coordinates in the second example



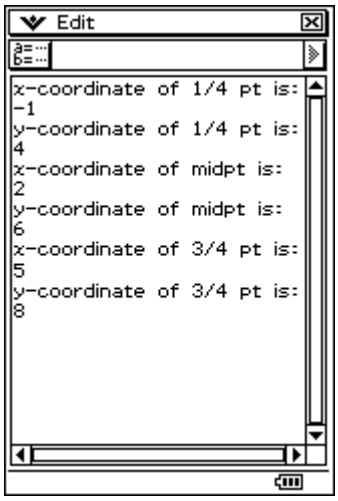


PART II
Practice Exercises

1. This practice set may not be easy, but it will definitely be interesting.
2. We know that the midpoint of a line segment divides the line segment into two equal parts. What if we want to divide it into four equal parts?
3. Modify your Midpoint program to find the ¼ point, the ½ point (this is already done – the midpoint) and the ¾ point.



4. After you have finished, run your program using the coordinates: (-4, 2) and (8, 10). Your output should look like:



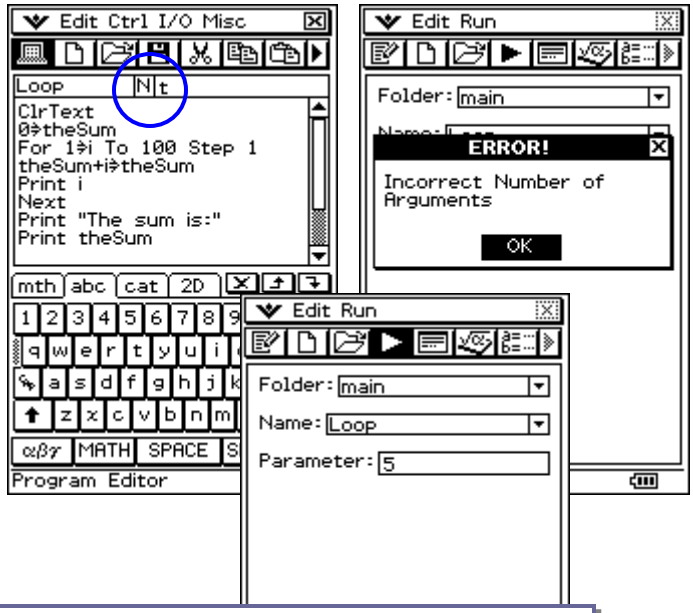
5. Run your program again using the coordinates: (2, -6) and (-8, 14).
6. With your output window resized, get a **screen capture.** Paste it into your Lesson 16 document (under a title of PART II).

PART III

In this part we will modify your Loop program to use a *parameter*. The *parameter* will require the user to input a value called an *argument* that will be stored in the *parameter* before running the program. We will also learn how to run programs from within Main and eActivity. This part is easy!

1. Adding a Parameter

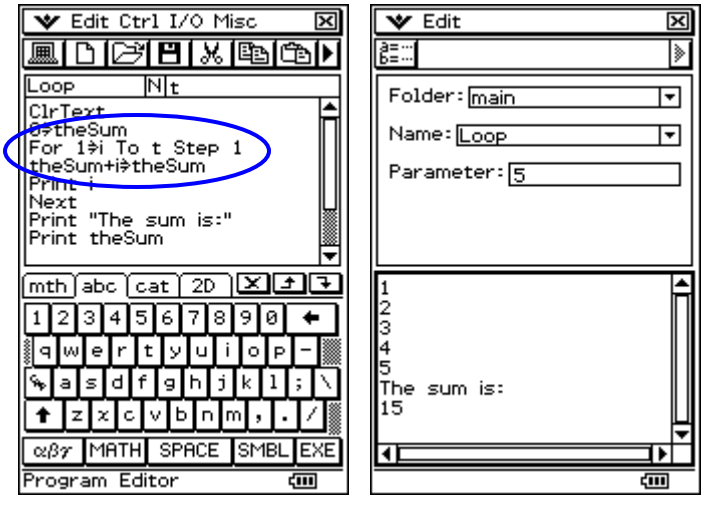
a. Click m and then p
b. Set Name: to **Loop**
c. Click P to open editor
d. Add a parameter by placing **t** following N
e. **Save** and **run** the program (not happy ☹)
f. Add an argument (any number) for our *parameter* **t**
g. **Run** the program again
h. It runs! The *parameter* **t** was happy to receive the number you provided.



Special Interesting Note:
Parameters store values temporarily. In my example, when I run the program **t** takes on a value of 5 until the program is finished. When finished, **t** returns to be whatever it was prior to running the program.

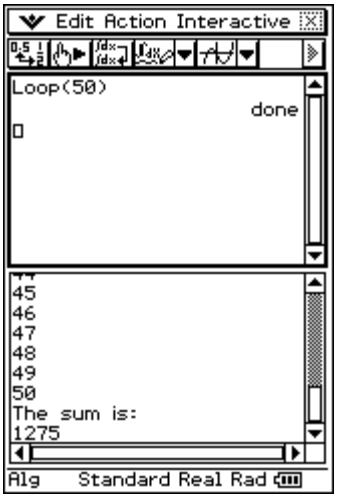
2. Using our Parameter to Control Looping

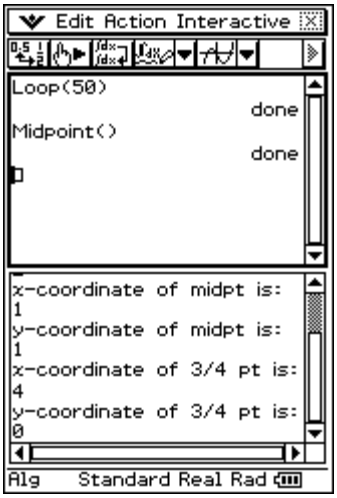
a. Click P to open editor
b. We will loop **t** times
c. Change the For To line to go up to **t**
d. **Save** your program
e. **Run** your program with a parameter value of 5



3. Running Programs from within Main

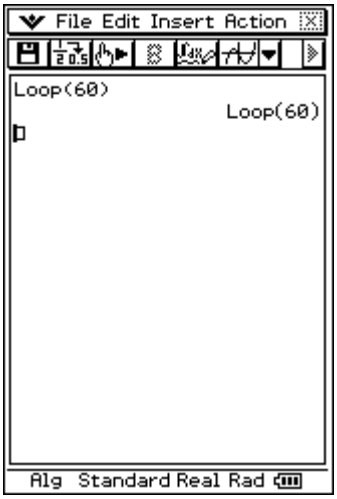
a. Click ⓜ and then J
b. Select **Edit/Clear All** (if needed)
c. Type in: **Loop(20)**
d. Press **EXE**
e. Change **20** to **50** and press **EXE** again
f. Type in: **Midpoint()**
g. Press **EXE**
h. Input data for Midpoint to complete the program run

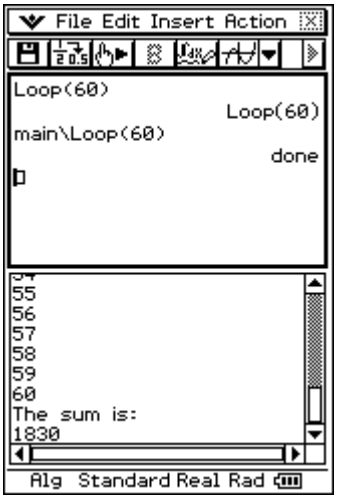




4. Running Programs from within eActivity

a. Click ⓜ and then A
b. Select **Edit/Clear All** (if needed)
c. Change to a Math line
d. Type in: **Loop(60)**
e. Press **EXE**
f. Hmm… eActivity needs help finding Loop
g. Type in: **main\Loop(60)**
h. Press **EXE**



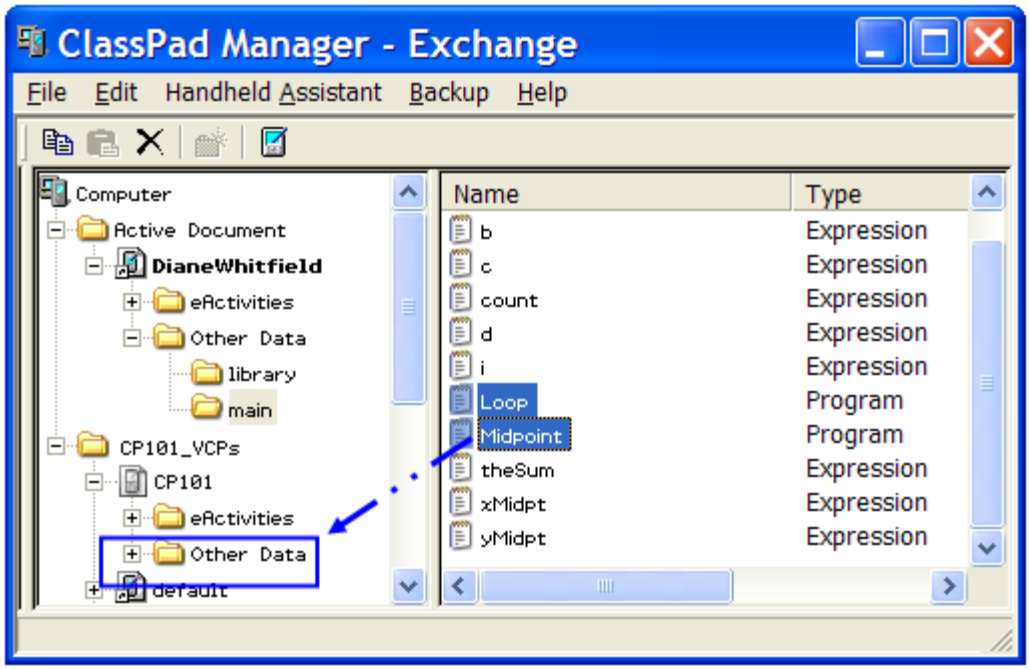


PART III
Practice Exercises

1. In this practice set, we will modify the Loop program to require two parameters. One will be used to begin the loop and the other to end the loop.
2. Open the Program application and then the Loop program for editing.
3. Change the Loop program to require two parameters by changing the **t** to **s,t**.
4. Edit the For To line to begin at **s** and end at **t** (i.e. start **i** at **s** instead of **1**).
5. Save and run your program using parameters of **3, 17**.
6. With your program output showing, get a **screen capture**. Paste it into your Lesson16 document (under a title of PART III).
7. Open the Main application.
8. Run your program using parameters of **-10, 5**.
9. With your program output showing, get a **screen capture**. Add two blank spaces following the first screen capture and then paste this one.
10. Open the eActivity application.
11. Run your program using parameters of **-10, 10**.
12. With your program output showing, get a **screen capture**. Add two blank spaces following the second screen capture and then paste this one.

Before continuing to Part IV, you can backup your Lesson 16 programs if you would like. To do this,

1. Please begin by opening the Exchange Window. [Hint: Right click on the ClassPad Manager emulator.]
2. **Click** on the **+** sign in front of the **Active Document** folder.
3. **Click** on **your name** so that its folders show.
4. Next, **double click** on the folder named **Other Data** and then **double click** on the **main** folder. We have a lot of variables defined!
5. **Click** on the **+** sign in front of the **ClassPad Manager** folder.
6. **Click** on the **+** sign in front of **CP101_Backup**.
7. Select your two programs. Hint: Select one, press the Ctrl key down and select the other.
8. Drag the selected program's to the **main** folder in **CP101** or another vcp file.



9. Or, you can drag and drop the entire main folder to the Other Data folder in the CP101 or another vcp file.

PART IV

Reflection Exercises

You have just completed the sixteenth lesson in ClassPad 101. In your spare time, please try writing other programs while the syntax is fresh in your mind. Please take a few moments to copy and paste the following three questions at the end of your Lesson16 document and answer them.

1. Approximately how long did it take you to complete this lesson?
2. Which activity did you enjoy the most?
3. Did you find any part of this activity difficult to follow? If so, which part? Also, how did you overcome the difficulty?

Note: To access the eActivities used by other lessons, you will need to activate the **CP101** vcp file again. Right click on the ClassPad Manager, select **Recent Documents** and then **CP101.vcp**.

Assessment 16: Introduction to Programming

- **Checkpoint**: Your word processed document, titled "Lesson16", should contain the following activities:
  1. Three screen captures from PART I.
  2. One screen capture from PART II.
  3. Three screen capture from PART III.
  4. Three reflection questions with answers from PART IV.
- **Submit** your **Lesson16 document** to your instructor for grading.

