Personal Polynomials

Teacher Notes & Answers

7 8 9 10 **11** 12

TI-Nspire™ Activity Teacher 4 hours

This document does not contain answers to the questions. To obtain a copy of the answer sheet and assessment rubric contact Texas Instruments via the Teacher Support email: teacher-support@list.ti.com

This document outlines reasoning behind the activity(s) associated with the Personal Polynomial Guided Exploration.

Teacher:

This is a guided exploration designed for Mathematical Methods Unit 1. The task is designed such that each student will develop a unique set of answers whilst essentially covering the same content. Each student develops a polynomial equation based on their name. Each letter in their name is assigned a number, based on alphabetical ordering: a = 1, b = 2 etc… The degree of the polynomial is therefore dependent on the student's name. Teachers are welcome to adjust the task as applicable to ensure greater equity, specifically for students with long names by using abbreviations, nick-names or simply their initials.

There are two opportunities to include programming in this exploration.

- Turning text into numbers
- Bisection method

Automating the process of turning text into numbers introduces students to a number of commands available on the calculator, includes content applicable to outcome 3 and opens up opportunities to automate the entire process of converting a name into a polynomial.

There are three options provided for the student document:

Student Document Version 1 – [7 pages]

- Printed instructions to help students in the programming components.
- QR Code for students to scan and watch instructional video for the programming component.
- The assessment rubric assumes the inclusion of the programming content.

Student Document Version 2 – [4 pages]

- QR Code for students to scan and watch instructional video for the programming component.
- The assessment rubric assumes the inclusion of the programming content.

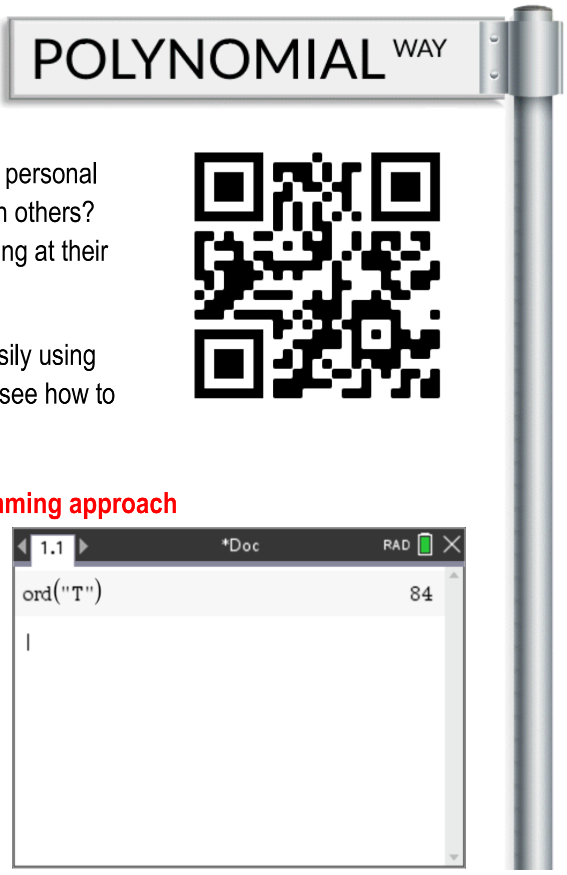
Student Document Version 3 – [2 pages]

- This version contains no programming and therefore removes several areas of Outcome 3.
- Reduced number of questions and estimated completion time.

For teachers there is a 'complete' version of the Personal Polynomial TI-Nspire file. This file generates the personal polynomial for any name making it quicker to mark student work! There is also an assessment rubric aligned to the corresponding areas of study and applicable outcomes as per Versions 1 and 2 of the student activity documents. Teachers need to make the necessary adjustments when using Version 3.

Author: Peter Fox

Introduction



Not everyone has a street named after them, but everyone can have a polynomial. What does your polynomial look like? In this activity you will change your name into a set of points and determine the polynomial that passes through these points, your personal polynomial. How does your polynomial compare and interact with others? How much can you tell about someone's polynomial by just looking at their name?

You can turn your name or any text into numbers quickly and easily using TI-Nspire. Scan the QR code or follow the instructions below to see how to write a program to automate the process.

Instructions – Converting your name into numbers – A programming approach

Start a new TI-Nspire document and insert a Calculator application.

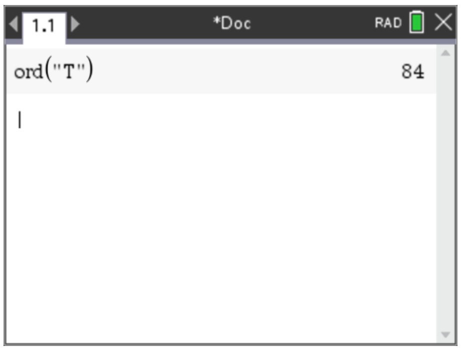
Type the command:

ord("T")

Next, type the command:

ord("t")

Notice the difference in values. Try other letters such as A and a.



```
ord("T")                                   84
```

Technology Tip!

The 'ord()' command on the calculator returns the ASCII code for the corresponding text. ASCII code maps 128 characters, the more sophisticated UNICODE maps 2^{21} characters.

Question: 1.

What value needs to be subtracted from capital letters to compute the value of each letter such that A = 1, B = 2 and so on?

Question: 2.

What value needs to be subtracted from lower case letters to compute the value of each letter such that a = 1, b = 2 and so on?

Comment: There is a great video on ASCII and UTF-8 to help understand the evolution and importance of both ASCII and UTF-8, definitely worth viewing: https://youtu.be/MijmeoH9LT4

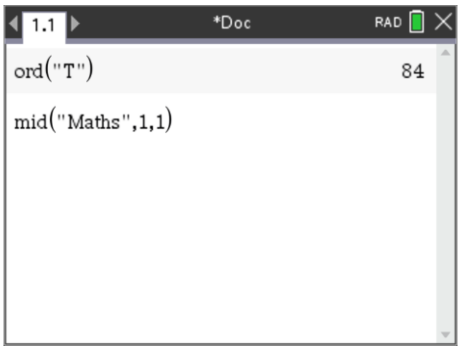
There are other useful text commands on the calculator that will prove useful.

Type the command:

mid("Maths",1,1)

Then try:

mid("Maths",2,1) and mid("Maths",3,1)



```
ord("T")                                   84
mid("Maths",1,1)
```

Author: Peter Fox

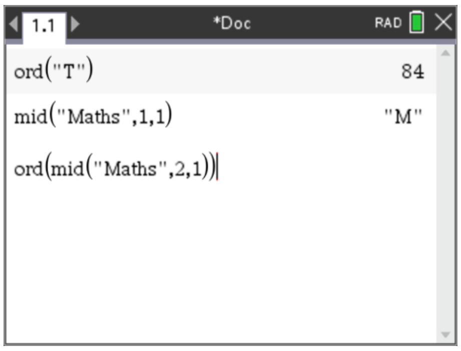
Question: 3.

Explain what the 'mid' command does.

Comment: There are several text based commands available on TI-Nspire that are very similar to those found in Excel. Commands like: Left, Right and Mid are available on Excel and TI-Nspire. The Ord command on TI-Nspire is equivalent to Code in Excel, Augment (TI-Nspire) is similar to Concatenate (Excel).

In addition to learning some clever text commands, this section of the guided exploration also encourages students to learn about some of the many in-built commands in TI-Nspire.

In mathematics one function can be embedded inside another, these are referred to as composite functions. The output of one function feeds directly into the input of another. This may be expressed as: $f(g(x))$.



```
ord("T")                    84
mid("Maths",1,1)           "M"
ord(mid("Maths",2,1))
```

In this case, two programming commands "ord" and "mid" will be combined to produce a single result.

The output from the 'mid' command is fed into the input of the 'ord' command.

Type in the following combination:

ord(mid("Maths",2,1))

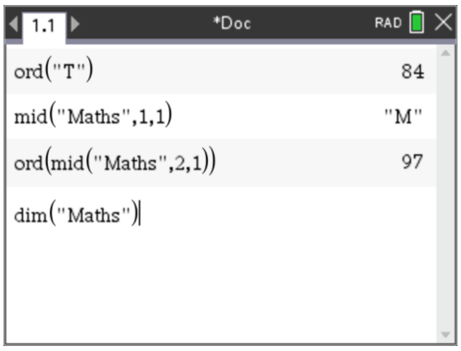
Technology Tip!

A short cut for entering the ord(mid("Maths",2,1)) combination is to start with ord(then arrow up through the history to select mid("Maths",2,1) and press **Enter** to paste it directly into the ord() command.

The last piece in this puzzle is the "dim" command.

Try the following:

dim("Maths")



```
ord("T")                    84
mid("Maths",1,1)           "M"
ord(mid("Maths",2,1))       97
dim("Maths")
```

Question: 4.

What does the dim() command do?

You're now ready to write a program!

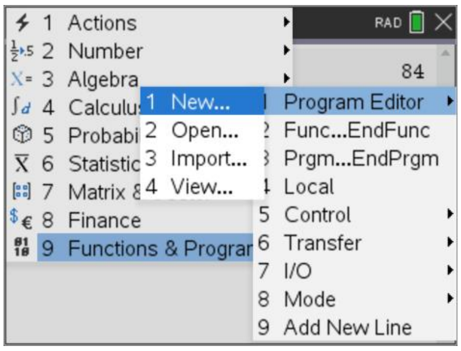
From the **menu** select:

Functions & Programs > **Program Editor** > **New**

Enter the name for your program:

N2N (name to numbers)

The program application will appear in a split screen. Press Ctrl + 6 to ungroup the two pages. The program editor will be moved to Page 1.2. Navigate to Page 1.2 to start working on the program.



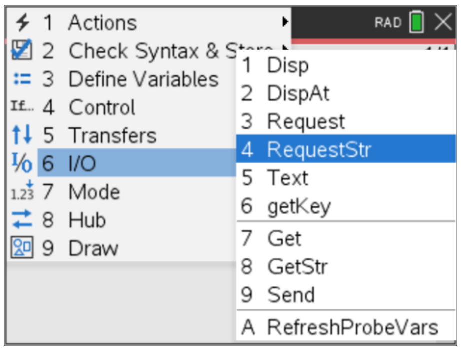
The first step in the program is to request the text to be converted into numbers. Text variables in programs are referred to as strings.

From the **menu** select

I/O > **RequestStr**

I/O = Input / Output

RequestStr = Request a string.

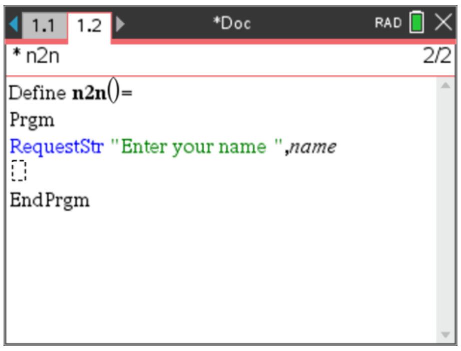


The Request-String command can include some prompting text relating to what is required.

Enter the following in the request command:

"Enter your name ",name

This creates a text prompt "Enter your name" and creates a 'string' variable 'name' that will hold the text to be converted into numbers and ultimately become a polynomial.



```
Define n2n()=
Prgm
RequestStr "Enter your name ",name

EndPrgm
```

The value assigned to each letter will be determined character by character, a loop needs to be created to extract each letter using the commands explored earlier. The number of times the loop needs to be executed is determined by the number of letters in the 'name'.

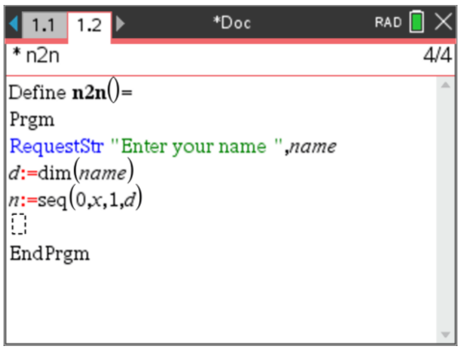
Type the following:

d:=dim(name)

A list also needs to be created to hold all the values.

n:=seq(0,x,1,d)

This creates a list filled with zeros, place holders for our final values.



```
Define n2n()=
Prgm
RequestStr "Enter your name ",name
d:=dim(name)
n:=seq(0,x,1,d)

EndPrgm
```

Author: Peter Fox

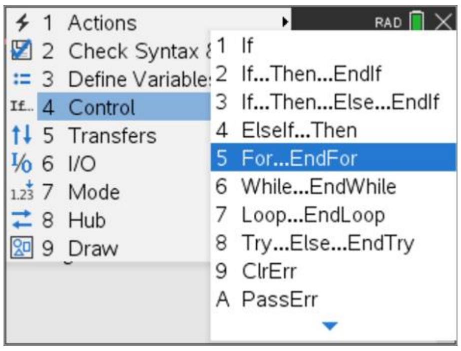
A "For" loop can be used to successively capture the value of each letter.

Use the **menu** and select:

Control > **For…EndFor**

Set the loop with a counter (i), starting at 1 and finishing at 'd'.

See below right for more detail on the syntax of the For command.

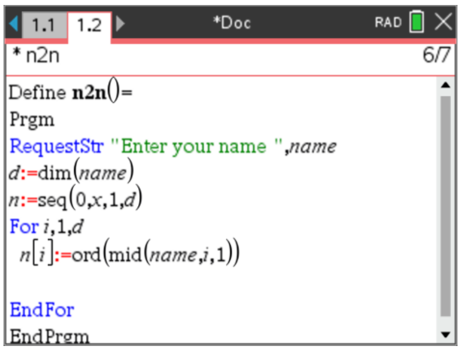


The first value in the list is equal to the numerical quantity aligned with the first letter of the 'name'. To identify the first value in the list for 'n', use [ ] brackets.

Type:

n[i]:=ord(mid(name,i,1))

Recall from the previous questions how the 'ord' and 'mid' commands work.



```
* n2n                                   6/7
Define n2n()=
Prgm
RequestStr "Enter your name ",name
d:=dim(name)
n:=seq(0,x,1,d)
For i,1,d
 n[i]:=ord(mid(name,i,1))

EndFor
EndPrgm
```

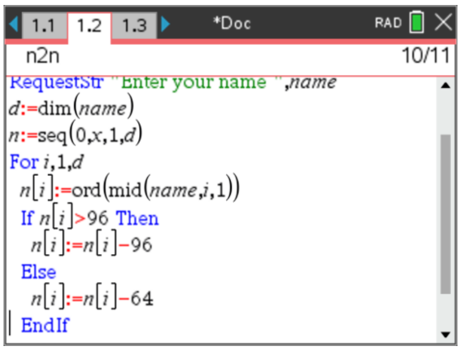
The final step in the program is to make the letter values range between 1 & 26. This can be done using an **IF** … **THEN** … **ELSE** statement.

If the letter value is greater than 96, then it is lower case. The ASCII code for 'a' is 97. The ASCII code for 'b' is 98 and so on, subtracting 96 will align these values to 1 to 26 for a to z respectively.

The ASCII codes for upper case letters such as "A" start at 65, so we need to subtract 64.

Notice how lower case letters were checked first! Why?

The program is now ready to run. Press **Ctrl** + **R**, then **Enter** to execute.



```
n2n                                    10/11
RequestStr "Enter your name ",name
d:=dim(name)
n:=seq(0,x,1,d)
For i,1,d
 n[i]:=ord(mid(name,i,1))
 If n[i]>96 Then
  n[i]:=n[i]-96
 Else
  n[i]:=n[i]-64
 EndIf
```

Question: 5.

Write down your name as numbers from 1 to 26. [Note: On the calculator you can type: n]

Comment: Of course students could have got to this point quicker by simply using an alphabet listing with the associated numbers; however there are so many opportunities for programming in Mathematics. This simple program does not assume any 'mathematical' knowledge, rather logical thinking!

Question: 6.

Write down the coordinates for the points that your polynomial must pass through: (1, #), (2, #) etc...

Question: 7.

What is smallest degree polynomial that will pass through all of your points? What assumptions do you need to make?

Question: 8.

Define your polynomial as: $f(x) = ax^n + bx^{n-1} + cx^{n-2} + \ldots$ according to the degree of your polynomial. Using your points from Question 6, write down the simultaneous equations that need to be solved in order to determine the equation for your polynomial.

Technology Tip!

If you used a program to help with changing your name from letters to numbers, you will need to delete the value stored in 'd': **DelVar** d

Teacher Notes:

To check the coefficients for each equation, run the mypoly() program in the Personal Polynomial TI-Nspire file. Use the student's name. A matrix "M" is created that contains the coefficients. Once the program has finished; type: M to see the coefficients. A copy of the TNS file will be provided with the Answers via email as per page one of this document.

Question: 9.

Determine the solution to the simultaneous equations (Question 8) and hence write down the equation to your polynomial.

Teacher Notes:

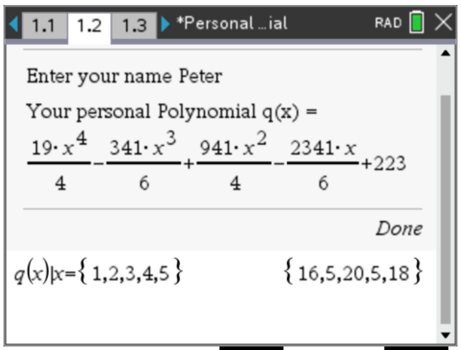
Students can work through the equations by hand depending on the degree of the polynomial, however this can generate a disproportionate amount of work for Alexandria compared with Joe. Students may also use the 'solve' command or set up the equations as a matrix and use Row-Reduction-Echelon form.

Question: 10.

Generate a table of values to show that your polynomial passes through the appropriate points.

Teacher Notes:

The teacher TNS file stores the student equation in q(x). It is therefore quick and easy to perform a substitution as shown opposite.



```
Enter your name Peter
Your personal Polynomial q(x) =
```

$$\frac{19\cdot x^4}{4}-\frac{341\cdot x^3}{6}+\frac{941\cdot x^2}{4}-\frac{2341\cdot x}{6}+223$$

```
                                    Done
q(x)|x={1,2,3,4,5}        {16,5,20,5,18}
```

Question: 11.

Use the bisection method to determine the x-intercepts of your polynomial.
[Consult with your teacher if your personal polynomial does not have any x axis intercepts.]

You can write a program to perform the bisection method. Use the instructions below or scan the QR code to watch an explanation of the bisection method and instructions for writing the program.

Teacher Notes:

The teacher TNS file includes the bisection program.

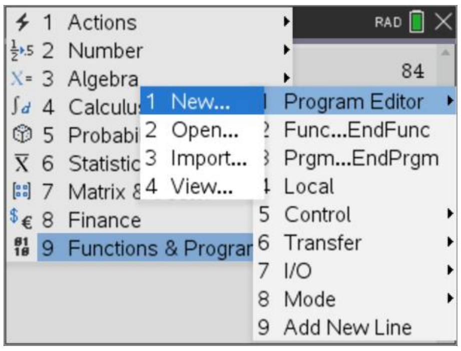
Where a student's polynomial does not cross the x – axis, students can find where their polynomial intersects with the line: y = 10 (or similar). This means the student needs to translate their polynomial accordingly and the use the bisection method on the result.

Bisection Method

You can use the bisection method to find where your polynomial crosses the x axis. For the purposes of these instructions, it is assumed that the polynomial is stored in: $f(x)$

Insert a new program and call the program:

Bisection



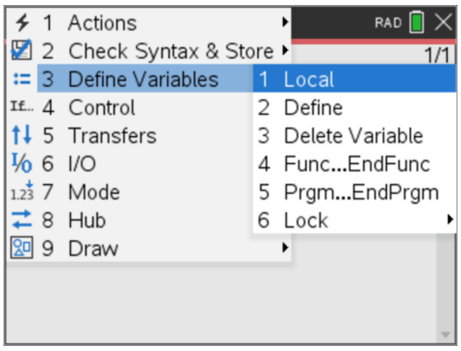
Author: Peter Fox

Some variables may have already been used resolving your personal polynomial. It is therefore worthwhile defining some of the variables used in this program as 'local'. Defining variables as local means they won't interfere with variables of the same name outside the program.

From the **menu** select:

Define Variables > **Local**

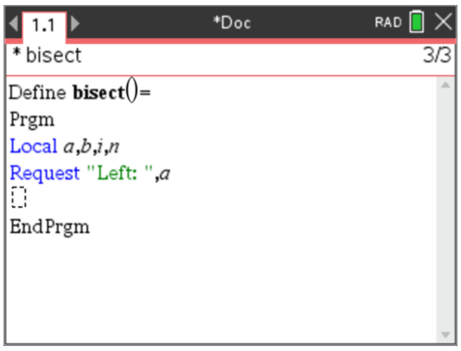
Variables a, b, i, n are all local. [Use a comma to separate variables]



The program needs a left and right bound location between which to search for solutions. So the first step is to request these values.
From the **menu** select:

I/O > **Request**

Enter the text in quotation marks: "Left: " followed by a comma and the variable 'a'. When the program is running the prompt "Left" will appear and the value will be stored in 'a'.



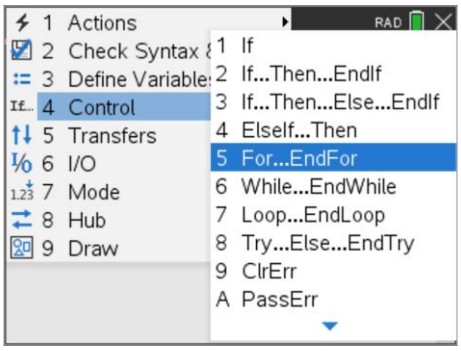
```
1.1    *Doc    RAD
* bisect                 3/3
Define bisect()=
Prgm
Local a,b,i,n
Request "Left: ",a

EndPrgm
```

Repeat this process and request the "Right" boundary and store it in 'b' and also the number of "Iterations" to be stored in 'n'.

The bisection method is an iterative process which means it repeatedly follows a set of steps to get closer and closer to the solution. A range of loop options are available, for this program a FOR loop will be used.

From the **menu** select:

Control > **For...EndFor**

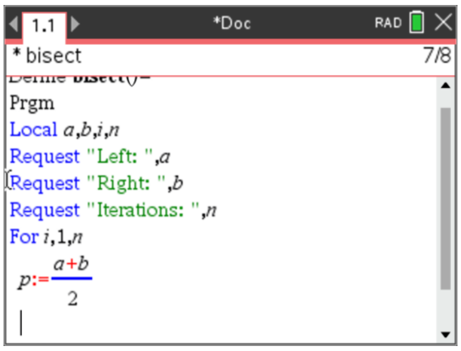


The accuracy of the result is reliant on how close a and b are to the axis intercept and also the number of iterations determined by the value of n. So the loop should be: **For i, 1, n**

Point P is located half way between 'a' and 'b'.

Assign the value of p such that:

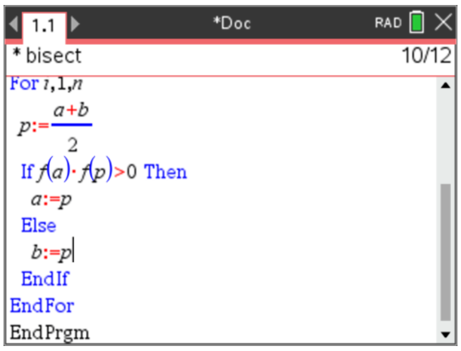
$$p := \frac{a+b}{2}$$



```
1.1    *Doc    RAD
* bisect                 7/8
Define bisect()=
Prgm
Local a,b,i,n
Request "Left: ",a
Request "Right: ",b
Request "Iterations: ",n
For i,1,n
  p:=(a+b)/2
```

It is now decision time! The x axis intercept lies somewhere between $x = a$ and $x = b$. The bisection method checks the midpoint (p) using the product of the y coordinates. If $f(a) \times f(p)$ is positive then points a and p must be on the same side of the x axis. This means the x axis intercept lies somewhere between p and b, otherwise it lies between a and p.

The IF … THEN … ELSE command is located in the **control** menu. Match the conditions and outcomes shown opposite and detailed above.



```
1.1    *Doc    RAD
* bisect                 10/12
For i,1,n
  p:=(a+b)/2
  If f(a)·f(p)>0 Then
    a:=p
  Else
    b:=p
  EndIf
EndFor
EndPrgm
```

The program is now ready to run, however to see what is happening a couple of Display At (dispat) commands are useful.

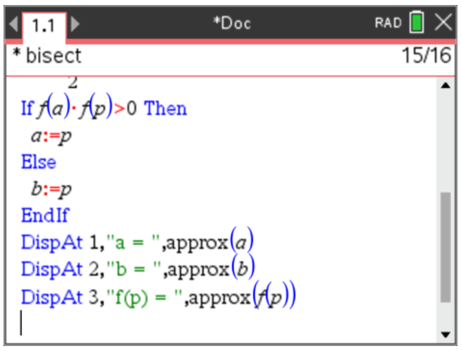
From the **menu** select:

I/O > **DispAt**

Each time the loop is executed, display the following information on lines one through to three:

Current value of ‘a’, ‘b’ and ‘$f(p)$’.

It is also worthwhile adding a wait command to make it easier to watch what is happening.



```
If f(a)·f(p)>0 Then
 a:=p
Else
 b:=p
EndIf
DispAt 1,"a = ",approx(a)
DispAt 2,"b = ",approx(b)
DispAt 3,"f(p) = ",approx(f(p))
```

Exploring more Personal Polynomials

Question: 12.

Hannah noticed that her polynomial had a smaller degree than she first thought. Explain why this is so.

Comment: It is possible for students to explain or justify their answer without the necessity to determine the polynomial. Teachers are encouraged to advise students as to whether or not they should include the polynomial definition. The degree of supporting evidence can help provide the distinction between student scores.

Question: 13.

Arora and Mayam both noticed that Elle and Hannah’s polynomials had a smaller degree than expected, but their polynomials were ‘as expected’. Explain why this is the case.

Question: 14.

Amy joins the conversation and states that her name is unique. It too has a lower than expected degree, but for a different reason. Explain. Can you find another name that also has this property?

Comment: A simple program can be written to systematically scan all names and display them on the screen momentarily. This however is not the intent, rather for students to think about the necessary criteria. This is relatively ‘straight’ forward for linear functions, but considerably more complicated for quadratics and higher degree polynomials.

Question: 15.

Which of the following names would provide polynomials $p(x)$ such that $p(x) = 0$ has at least one solution:

i) Daisy   ii) Peter   iii) Steven   iv) Emma   v)John

Question: 16.

Bindi notices that her polynomial $p(x)$ has no solutions such that $p(x) = 0$. Find some other names that have no solutions to $p(x) = 0$. Explain how you searched and what criteria you used.

Question: 17.

According to their polynomials, where do Lilly and Lillian meet?

Question: 18.

Victor and his friend Victoria want to explore all the places where there polynomials meet. Without determining their equations:

i) Identify six points where their polynomials will intersect.

ii) Explain why their polynomials have seven points of intersection.

Question: 19.

Anna and Hannah claim to have very similar polynomials. Apply an appropriate transformation and compare the two polynomials.

Question: 20.

Are there any two names that share the same polynomial? Explain your reasoning.

Comment: At this point in the Guided Exploration, students should have sufficient skills and knowledge to 'explore' without additional scaffolding. This question is another area where student's unique explorations and depth of exploration can differentiate between student skills, knowledge and effort.