Coding Hail Stones

Teachers Teaching with Technology™
Professional Development from Texas Instruments

Student Activity

7 **8** 9 10 11 12

TI-Nspire Coding Student 120 min

Introduction

The Hail Stone sequence is produced by a relatively simple formula. What makes this sequence important and popular is Collatz's conjecture: "Start with any number and the sequence will eventually produce a one (1)." This statement by Collatz is referred to as a conjecture because it has never been proven. Trying lots of numbers doesn't prove the statement, it simply shows that all the numbers tested so far end up with a one. Finding a sequence that seems to go on for ever doesn't prove Collatz wrong, unless it has produced a loop. Really long sequences are exactly that, really long. His conjecture could be proved using a mixture of number theory and algebra, but despite numerous efforts, no one has succeeded in such a proof. So what are we looking for in this investigation? Patterns, generating ideas based on observable patterns. A powerful problem solving technique is to throw some numbers into the mix and pay careful attention to what comes out.

So why is the conjecture referred to as "The Hailstone Problem"? Hail stones are formed when high level winds carry water droplets upwards where the air temperature is cold enough to freeze water. The water droplets become frozen, as they continue to go up and down through these high altitudes they get bigger and heavier, and eventually fall to the ground. The numbers in the sequence go up and down, and according to Collatz, will eventually come tumbling down to one!

The Rule

Start with any number:

• If it is odd, multiply it by three then add one.	Odd … x 3 + 1
• If it is even, divide by two.	Even … ÷ 2

Repeat this odd/even process until you reach 'one'.

Example:

Starting Number:	11
Since 11 is odd:	11 x 3 + 1 = 34
Since 34 is even:	34 ÷ 2 = 17
Since 17 is odd:	17 x 3 + 1 = 52
Since 52 is even:	52 ÷ 2 = 26

Question: 1.

Write down the remaining numbers in the sequence until one (1) is reached.

Author: P. Fox

Question: 2.

Write down the sequence for the set of numbers starting with 3.

Question: 3.

Without making any more calculations, write down the sequence for the set of numbers starting with 5. Explain how you were able to write this sequence down without any further calculations.

Question: 4.

Write down the sequence for the numbers listed in the table below.
(Some of the sequences have already been completed for you.)

Start	Sequence	Number of Terms
6	3, 10, 5, 16, 8, 4, 2, 1	9
48		
13		
7		
9		
29	88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1	
53		

Question: 5.

Some sequences are surprisingly short; others can be extraordinarily long, particularly considering their starting value. The sequence for 53 contains just 12 terms, the sequence for 54 contains 113. How many terms would be in the sequence for the starting value of 864?

Automating the Process

Testing numbers can be a very time consuming process. In this investigation we are interested in exploring patterns that may help explain why the sequence seems to always end at one. Computers are very quick at making programmed decisions and performing calculations. In this section you will write a computer (calculator) program to perform the calculations very rapidly. A few commands must be learnt before writing the program. The following questions will help understand these commands.

Question: 6.

Record the result for each of the following calculations. The MOD command can be retrieved from the **Number** menu in the calculator application, followed by **Number Tools** then **Mod**; alternatively the word Mod can be typed directly using the keypad. Once the command has been used it can be copied (Ctrl + C) and pasted (Ctrl + V) … and re-pasted (Ctrl + V) for each subsequently use.

a) Mod(25,2)

b) Mod(13,2)

c) Mod(12, 2)

d) Mod(40,2)

e) Mod(9,2)

f) Mod(100, 2)

g) Mod(1,2)

h) Mod(1008,2)

i) Mod(147, 2)

Author: P.Fox

Question: 7.

Explain what Mod(*n*,2) does in terms of the number *n*.

If you are completely new to programming, try the introductory programming lessons on the Texas Instruments website: 10 minutes of coding.

https://education.ti.com/en-GB/aus-nz/product-resources/ti-codes

Programming

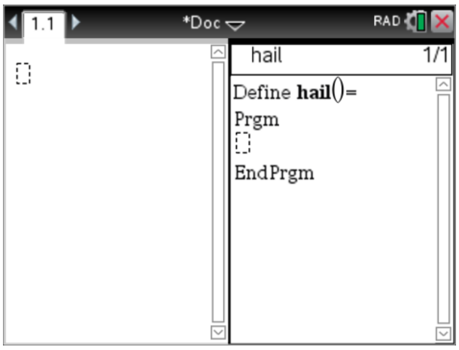
The entire program contains less than 10 lines of code. The real decision making process occurs with the testing statement “IF … THEN … ELSE”.

Request n	To test a single number (n) the user will be prompted to input a number. The ‘Request’ command seeks an input.
WHILE n > 1	This line assumes the sequence will eventually end up at one, so the instructions are repeated while n is bigger than 1.
IF n is odd **THEN**	Decision time, need to automatically determine if a number is odd. To achieve this: Mod(n,2) will be used.
n := 3 x n + 1	This is not an algebraic statement. If n is odd, value of n changes according to the rule. “ : = “ is read as ‘assigned equal to’.
Else	If the number is not odd… it must be even.
n : = n ÷ 2	This is not an algebraic statement. The value of n changes according to the rule for even numbers.
Endif	Programming command representing the end of the decision making process.
Disp n	For the benefit of the user, the new value of n is displayed so that it is possible to see the entire sequence.
EndWhile	This is the end of the loop that performs the calculations, and will also be the end of the program once n reaches 1.

On the calculator application, use the **[Menu]** key then select **Functions & Programs**, followed by **Program Editor** then **New**.

Type the program name: Hail

Press [**Enter**] then arrow down to the first space in the program.

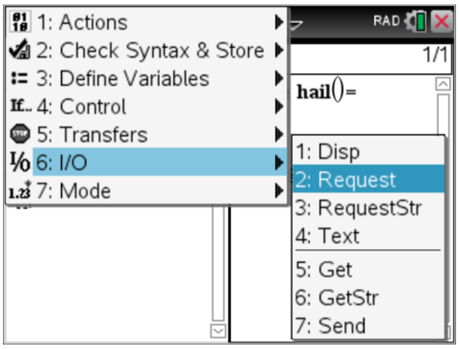


```
Define hail()=
Prgm

EndPrgm
```

Author: P.Fox

The first step in the program is to request an input value. From the **[Menu]** select **I/O** (Input/Output) followed by **Request**. The request is best supported with a word prompt.

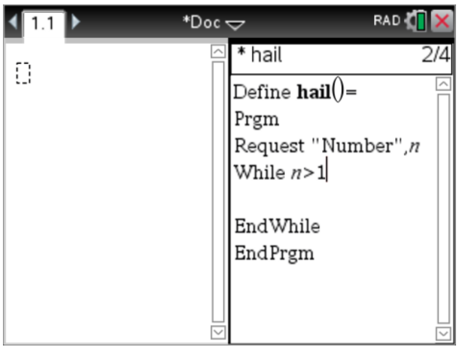
Type: "Number" , n



```
hail()=
```

The quotation marks can be accessed by selecting the [**?!**] key. The variable *n* is short for number and will hold the initial number, but will change as soon as the program executes. The second step in the program is to start a loop which will continue while the value of *n* is greater than 1. From the **[Menu]** select **Control** followed by **While …EndWhile**.

Type: n > 1



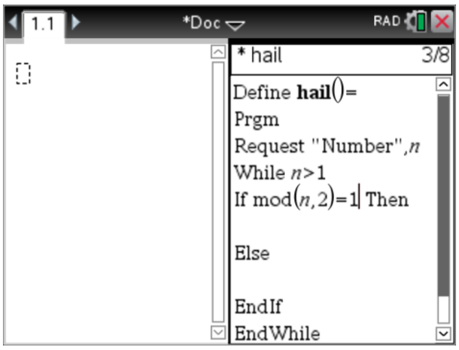
```
Define hail()=
Prgm
Request "Number",n
While n>1

EndWhile
EndPrgm
```

The '>' sign is located 'under' the [=] key and can be accessed by pressing Ctrl + =

At this point the decision algorithm is involved. The command Mod(n,2) will return a value of 1 if n is an odd number. From the [**Menu**] select **Control** followed by **IF … THEN … ELSE … ENDIF**.

Between the words "IF" and "Then" type:

Mod(n,2)=1



```
Define hail()=
Prgm
Request "Number",n
While n>1
If mod(n,2)=1 Then

Else

EndIf
EndWhile
```

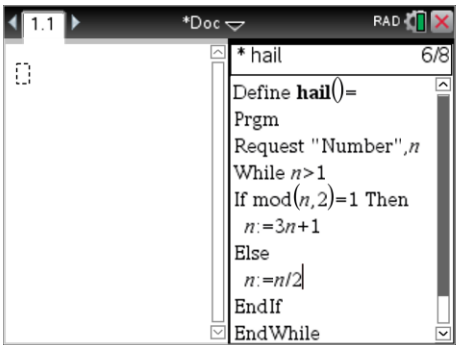
If Mod(n,2)=1 Then the value of n must be multiplied by 3 and one added to the result … and then stored back in *n*.

To achieve this result, after the word "Then" , press [**Enter**] to start a new line, then type:

n := 3*n* + 1

After the word "Else", press [Enter] to start a new line then type:

n := n / 2



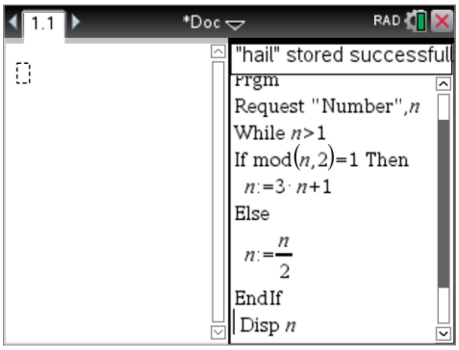
```
Define hail()=
Prgm
Request "Number",n
While n>1
If mod(n,2)=1 Then
  n:=3n+1
Else
  n:=n/2
EndIf
EndWhile
```

Move the cursor to a line between Endif and Endwhile, if one doesn't exist, place the cursor after Endif and press [**Enter**]. The program needs to show the progress of the sequence now that n has been changed.

From the **I/O** menu select **Disp** and then type n.

Disp n

Once this is done, press **Ctrl** + **B** to store the program. Your program is now ready. Press **Ctrl** + **Tab** to shift focus back to the calculator application, away from the programming tool.

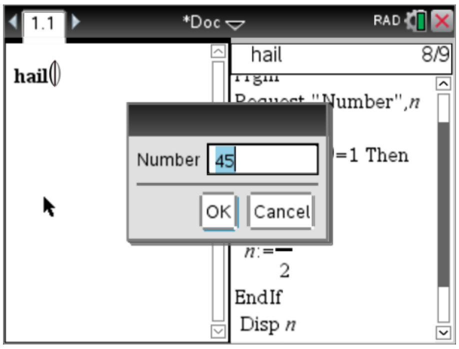


```
"hail" stored successful
Prgm
Request "Number",n
While n>1
If mod(n,2)=1 Then
  n:=3·n+1
Else
  n:=n/2
EndIf
Disp n
```

Author: P.Fox

Press the [**Var**] key and select the “Hail” program.
The text Hail() will appear on the screen, press [**Enter**] to run the program and enter a number, start small so you don’t get ‘too much hail’.

To review the numbers in the sequence use the up / down and left / right arrows to scroll through the sequence.



```
hail()
```

```
hail                          8/9
Prgm
Request "Number",n
Number 45
=1 Then
n:=n/2
EndIf
Disp n
```

Question: 8.

Use your program to determine how many numbers are in the ‘45’ sequence?

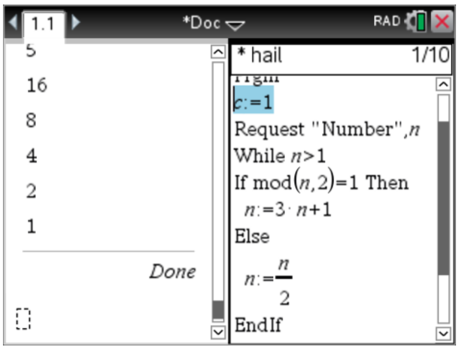
Question: 9.

Use your program to determine how many numbers are in the ‘47’ sequence … and comment on any short comings of the program.

A counting system can be incorporated into the program to save scrolling. To shift focus back to the program press **Ctrl** + **Tab**.
Insert a line between Prgm and Request. A ‘counter’ should always be reset at the start of a program to ensure it is not adding on to a previously stored value.

c : = 1

The counter will start with ‘1’ representing the start number.

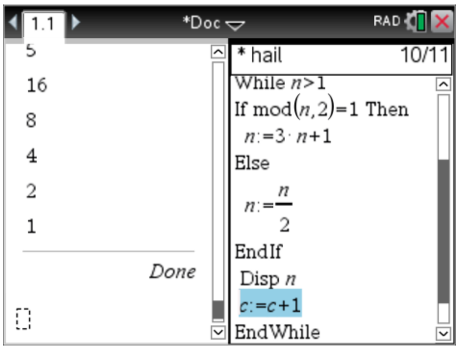


```
5
16
8
4
2
1
        Done
```

```
* hail                        1/10
Prgm
c:=1
Request "Number",n
While n>1
If mod(n,2)=1 Then
 n:=3·n+1
Else
 n:=n/2
EndIf
```

Enter a line between Disp and EndWhile, this is where the counter will be increased every time the loop makes another decision, odd or even.

c : = c + 1



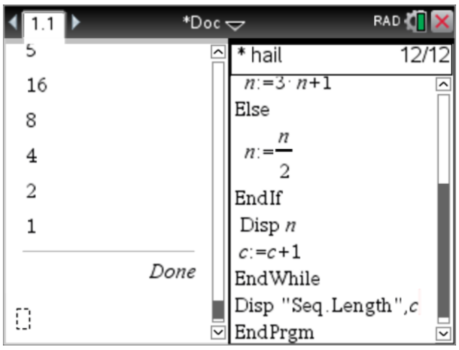
```
5
16
8
4
2
1
        Done
```

```
* hail                        10/11
While n>1
If mod(n,2)=1 Then
 n:=3·n+1
Else
 n:=n/2
EndIf
 Disp n
 c:=c+1
EndWhile
```

One final instruction, for the user, when the loop has finished the quantity of numbers in the sequence needs to be displayed.
Between EndWhile and EndProgram type the line:

Disp “Seq. Length “,c

Save the changes to the program by pressing **Ctrl** + **B**.
Change focus back to the calculator, **Ctrl** + **Tab.**
Try some of the numbers from Question 4 to check that your program is working before continuing.



```
5
16
8
4
2
1
        Done
```

```
* hail                        12/12
 n:=3·n+1
Else
 n:=n/2
EndIf
 Disp n
 c:=c+1
EndWhile
Disp "Seq.Length",c
EndPrgm
```

Question: 10.

Challenge your friends to see which number(s) between 1 and 100 produce the longest sequence? Record your research and findings.

Author: P.Fox

Exploration

Mathematicians use patterns and logic to explore. It's your turn to explore the Hail Stone problem. A few starting ideas are presented here, choose one of these or discuss your own idea with your teacher.

- Graph sequence length versus starting number and discuss any patterns you find.
- Explore hail stone sequences in terms of odds and evens and discuss/explain any patterns you find.
- John says "Since 50% of whole numbers are odd, and multiplying by 3 is more 'powerful' than dividing by 2, sequences should get bigger not smaller." Discuss John's statement, supporting your comments with numerical (or algebraic) evidence.
- Prime numbers are used in many areas such as encryption, because of their 'lack' of factors. Explore the average length of sequences produced by prime numbers and compare with the sequence length of composite numbers.
- Write your own inquiry brief. (Confirm your brief with your teacher.)