ULAM’S CONJECTURE

This topic illustrates *How to Invent an Algorithm* and *Help for Beginners*.

**Example**
Start with any positive integer. If it is even, divide it by two; if it is odd, multiply it by three and add one. Obtain successive integers by repeating this process. The Polish-American mathematician Stanisław Ulam (1909–1984) conjectured that eventually the number 1 will be obtained no matter what the starting integer is.

### How to Invent an Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
</table>
| 1    | Understand the problem by solving it | • Use pencil and paper to solve the problem by hand.  
• Use George Pólya’s problem solving techniques.  
• Write a list of any questions &or ambiguities you encounter and seek answers. |

Solve Ulam’s conjecture for input of 40:

40  20  10  5  16  8  4  2  1

For an input of 17:

17  52  26  13  40

For an input of 7:

7  22  11  34  17
Questions and Answers

Q: What should I do with an input of 1? Should I print 1 and stop or should I print 1, 4, 2 and 1
A: Print 1, 4, 2 and 1

Q: What is a “positive integer”?
A: A whole number that is greater than 0.

Q: Should my program check for that?
A: It should and since it’s easy to do we’ll require that.

How to Invent an Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
</table>
| 2    | Determine the program’s requirements | • Identify the program’s input and output.  
• Determine what computations transform the input into the output. |

INPUT

PROCESSING

OUTPUT

integer → Calculate each successive integer in the Ulam sequence → each integer
### How to Invent an Algorithm – Help for Beginners

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
</table>
| 3    | Use top-down design to develop a pseudo-code algorithm that solves the problem | • Write down a first-draft algorithm as follows:  
  o Write in English the steps you employed when solving the problem by hand.  
  o Don't worry about language or details; just get your ideas on paper in outline form.  
  o Be sure to include the input and output steps. |

Returning to the pencil and paper work in step 1, write the steps employed:

**Ulam’s Algorithm (1st draft)**

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
</table>
| 3    | Use top-down design to develop a pseudo-code algorithm that solves the problem | • Write down a first-draft algorithm as follows:  
  o Write in English the steps you employed when solving the problem by hand.  
  o Don't worry about language or details; just get your ideas on paper in outline form.  
  o Be sure to include the input and output steps. |

Returning to the pencil and paper work in step 1, write the steps employed:

**Ulam’s Algorithm (1st draft)**

input integer 40 and print it  
40 is even so divide it by 2 to get 20 and print it  
20 is even so divide it by 2 to get 10 and print it  
10 is even so divide it by 2 to get 5 and print it  
5 is odd so multiply it by 3 and add 1 to get 16 and print it  
16 is even so divide it by 2 to get to get 8 and print it  
8 is even so divide it by 2 to get to get 4 and print it  
4 is even so divide it by 2 to get to get 2 and print it  
2 is even so divide it by 2 to get to get 1 and print it
### How to Invent an Algorithm – Help for Beginners

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
</table>
| 3    | Use top-down design to develop a pseudo-code algorithm that solves the problem | - Introduce program variables to hold data you’ve identified.  
- Identify and clarify any loops that you need:  
  o In your draft outline, look for (1) verbs that imply repetition (e.g. repeat) and (2) repetitive sentences.  
  o To determine the loop’s truth value, look for variables that change inside the loop and ask what they should be when the loop is to quit.  
  o Make appropriate initializations before the loop.  
  o Isolate these sentences and wrap them within an appropriate pseudo-code looping statement.  
- Identify and clarify any selections that you need:  
  o In your draft outline, look for (1) conjunctions (e.g. if, when) and (2) alternate computations.  
  o Isolate the computations and wrap them within an appropriate pseudo-code selection statement.  
- Rewrite your algorithm accordingly. |

---

**Ulam’s Algorithm (1st Draft)**

```
input integer 40 and print it

40 is even so divide it by 2 to get 20 and print it
20 is even so divide it by 2 to get 10 and print it
10 is even so divide it by 2 to get 5 and print it
5 is odd so multiply it by 3 and add 1 to get 16 and print it
16 is even so divide it by 2 to get 8 and print it
8 is even so divide it by 2 to get 4 and print it
4 is even so divide it by 2 to get 2 and print it
2 is even so divide it by 2 to get 1 and print it
```

Selection based on `n` being even or odd

Loop quit when `n = 1`
The algorithm is rewritten using appropriate loop and selection statements. Check the questions and answers given earlier. To satisfy the first answer the algorithm starts the loop before checking that \( n \) is 1 so that an input of 1 doesn’t quit right away. To satisfy the second and third answers, the algorithm uses a selection statement to insure that the input is positive.

```
Ulam's Algorithm (2nd Draft)

input n
if n < 1 then
    complain
else
    repeat
        print n
        if n is even then
            n = n/2
        else
            n = n*3 + 1
        end if
    until n = 1
    print 1
end if
```
### How to Invent an Algorithm – Help for Beginners

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
</table>
| 4    | Check that your algorithm is correct and complete | • Desk check your loops:  
  o Work through the first few cycles of the loop.  
    ▪ Does each cycle bring the computation one step closer to the final desired result?  
    ▪ Does each cycle bring the truth value one step closer to loop termination?  
    ▪ When the loop quits is the final result achieved?  
  o Check for an off-by-one error, where the loop cycles one time too many or too few.  
  o Check that the algorithm works even if the loop never cycles.  
• Desk check your selections for boundary errors (e.g. \( x < y \) when it should be \( x \leq y \)). |

#### Desk Check of Ulam’s Algorithm (2nd Draft)

```
input n
if n < 1 then
  complain
else
  repeat
    print n
    if n is even then
      n = n/2
    else
      n = n*3 + 1
    end if
  until n = 1
  print 1
end if
```

<table>
<thead>
<tr>
<th>n's value</th>
<th>20</th>
<th>20</th>
<th>10</th>
<th>5</th>
<th>16</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>loop cycle</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>etc.</td>
<td></td>
</tr>
<tr>
<td>quits when n = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>