Your mission, should you choose to accept it, is to develop algorithms to complete tasks involving the relative order of $n$ distinct integers. The problem is that the integers are random and unknown to you! All you know is that they are named $a_1, a_2, \ldots, a_n$. Note that we call $i$ the index of the integer $a_i$.

For each task, your approach must work no matter what the order of the integers is.

Some good news: A helpful machine is available. The machine knows the relative order of these integers. To use it, you enter the indices of two integers into the machine and it will tell you which of the two corresponding integers is larger. For example, if $a_4 = 5$, $a_2 = 7$ and you enter 4 and 2 into the machine, it will tell you that $a_2$ is larger. We name the machine $M$ and in this case we have $M(4, 2) = 2$ and $M(2, 4) = 2$. Either of these application of $M$ tells you that the integer with index 2 is larger than the integer with index 4.

Some bad news: For each task, there is a limit on the number of times you can use the machine. This limit applies no matter what the relative order of the $n$ integers happens to be.

Some more good news: Your memory is perfect and you can remember (or record) the result every time you use the machine.

Example

Suppose $n = 4$ and you want to determine which of the integers, $a_1, a_2, a_3,$ or $a_4$, is the largest, while limiting yourself to only 3 uses of the machine. Here is one way to do this:

1. Compute $M(1, 2)$ and record this answer as the index $x$.
2. Compute $M(3, 4)$ and record this answer as the index $y$.
3. The largest integer is the integer with index $M(x, y)$.

The Tasks and a Fun Tool

Below are the three different tasks to be completed. Each task outlines how many integers there will be in the list ($n$), what you are attempting to answer about the list (Task), and how many times you can ask the machine for help (Limit).

<table>
<thead>
<tr>
<th>$n$</th>
<th>Task</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Determine the largest integer.</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>Determine both the largest integer and the smallest integer.</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>Determine the second largest integer.</td>
<td>9</td>
</tr>
</tbody>
</table>

Important: We have written a Python computer program that will generate random integers and simulate the helpful machine. It is a lot of fun to use this interactive tool to test if your solutions are correct. See the next page for instructions on how to use the tool.
Using the Tool

The tool works by repeatedly asking you what you want to enter into the machine and then displaying the result. After the number of times you have used the machine reaches the limit, it will ask you for the index of the largest integer in the list. It will then tell you whether or not you are correct.

You do not need to know anything about Python in order to use the tool.

Getting the correct answer for a few lists does not mean that you have a correct algorithm for the task.

Your algorithm has to work for any choice of integers, regardless of their order. The more you test your algorithm, the more evidence you have that it is correct. After testing out your algorithm using the tool, try to explain why your algorithm will work on all possible lists.

Here are instructions for using the tool:

1. Open this webpage in one tab of your internet browser. You should see Python code.
2. Open this free online Python interpreter in another tab. You should see a middle panel labelled main.py.
3. Copy the code and paste it into the middle panel of the interpreter.
4. Hit run. You will interact with the tool using the right black panel, and you might want to widen this panel.
5. After completing a test, or if you encounter an error, you can hit run to begin another test. If you want to start over during a test, you can hit stop and then run.

More Info:

Check out the CEMC at Home webpage on Wednesday, April 29 for solutions to the three tasks. Our webpage Computer Science and Learning to Program is the best place to find the CEMC's computer science resources. Two resources through which you can explore Python further are:

Python from scratch
A gentle introduction to programming designed with the beginner in mind.

CS Circles
Interactive lessons teaching the basics of writing computer programs in Python. This is also an introduction but moves at a bit of a faster pace.