Homework Rubric Examples

Rubric

Your homework problems will be graded according to this rubric. Each problem will be put into one of the following six categories and assigned the corresponding grade.

Correct and clear (4) Answers in this category will give a completely correct solution to the problem and present it in a clear, logical way

Correct and unclear (3) Answers in this category are correct, but presented in a way that is hard to follow or imprecise

Clear, but has a small mistake or two (3) Answers in this category are almost correct, but have an algebra mistake or some small error. The answer is presented clearly.

Partially correct (2) Answers in this category are along the right lines, but have some major flaws

Demonstrated some understanding (1) Answers in this category demonstrate that the student understood what needed to be done to solve the problem and made some headway, but didn't get to a solution

"This ain't it" (0) The student didn't try, or obviously doesn't understand what (s)he needed to do

Examples

The TA's have written examples of answers to the following problem that would fall into each category.

Problem

Write pseudocode for `array_max(array)` and explain why the runtime is linear.

Answers

Correct and Clear:

```
function array_max(array):
  """an array of integers -> An integer
  Purpose: To determine the maximum value of the given array
  Example: array_max([0 2 5 3 4] -> 5
           array_max([]) -> null
  """
  if length of array is 0:
    return null
  else
    m <- array[0]
    for i from 1 to the end of the array
      m <- max(m,array[i])
    return m
```

The runtime of this function is linear. The number of operations outside of the for loop is constant, and there is also a constant number of operations within the for loop. The for loop iterates through the entire array a single time, so this takes a total of $n \times k$ time, where n is the size of the array, and k represents a constant. Since constants are not used in determining running time, this simplifies to $O(n)$ time.

Explanation: The pseudocode follows standards and is easy to read. The running time explanation is clear. Nothing is repeated, sentences are short and meaningful.

Correct and not clear:

```
function array_max(array):
  if length of the array is 0:
    return null
  else if length of the array is 1:
    return array[0]
  else:
    max <- array[0]
    for elements in array:
      if max < array[i]:
        max <- array[i]
    return max
```

The runtime is defined in terms of how we loop. In this case we loop over all of the elements. Since we have n elements the runtime is n. Other operations, such as the if statements, are not taken into account because they do not fall within the loop.

Explanation: The pseudocode is correct but there are some lines that are less clear or do not conform to standards. The explanation is vague and a little bit imprecise.

Small math mistake:

```
function array_max(array):
  if length of the array is 0:
    return null
  else if length of the array is 1:
    return array[0]
  else:
    max <- array[0]
    for elements in array:
      if max > array[i]
        max <- array[i]
    return max
```

The runtime of this function is linear. The number of operations outside of the for loop is constant, and there is also a constant number of operations within the for loop. The for loop iterates through the entire array a single time, so this takes a total of $n \times k$ time, where n is the size of the array, and k represents a constant. Since constants are not used in determining running time, this simplifies to $O(n)$ time.

Explanation: The student has made the slight mistake of putting a $>$ instead of a $<$ in the pseudocode.

Partially correct:

```
function array_max(array):
  m <- array[0]
  for i from 1 to the end of the array
    m <- max(m,array[i])
  return m
```

The runtime of this function is linear. The number of operations outside of the for loop is constant, and there is also a constant number of operations within the for loop. The for loop iterates through the entire array a single time, so this takes a total of $n \times k$ time, where n is the size of the array, and k represents a constant. Since constants are not used in determining running time, this simplifies to $O(n)$ time.

Explanation: The student has a good explanation of the running time of the algorithm but seems to have misunderstood exactly the function that they should be writing. Still, they have the right idea of comparing values in the array to a maximum.

Demonstrated some understanding:

```
function array_max(array):
  m -> array[0]
  for i from 0 to the end of the array
    max -> true
    for j from 0 to the end of the array
      if array[j] > array[i]
        max -> false
    if max
      m -> array[i]
      break
  return m
```

The runtime of this function is quadratic, since I was not sure how it could be linear. The number of operations outside of the for loop is constant, and there is also a constant number of operations within the inner for loop. The for loops iterate through the entire array a single time each, so this takes a total of $n \times n \times k$ time, where n is the size of the array, and k represents a constant. Since constants are not used in determining running time, this simplifies to $O(n^2)$ time.

Explanation: The student was unable to come up with a linear time algorithm but their algorithm does produce a correct answer.

Didn't try/"This ain't it"

```
function array_max(array):
  max <- array[0]
  for i from 0 to the end of the array
    if max < array[i]
      return false
  return true
```

$O(n)$ time.

Explanation: The algorithm doesn't produce a correct answer and there is no explanation for the running time at all.