Congratulations! You passed!
TO PASS 80% or higher

Keep Learning

GRADE
100%

Mergesort Algorithm

LATEST SUBMISSION GRADE
100%

1. Consider the merge algorithm for merging two sorted arrays into one single sorted array.

5 / 5 points

```
1  def merge(a, b):
2      n = len(a)
3      m = len(b)
4      i = 0
5      j = 0
6      c = []
7      while ( i < n and j < m):
8          if a[i] <= b[j]:
9              c.append(a[i])
10             i = i + 1
11         else:
12             c.append(b[j])
13             j = j + 1
14     while i < n:
15         c.append(a[i])
16         i = i + 1
17     while j < m:
18         c.append(b[j])
19         j = j + 1
20     return c
```

Select all the correct facts from the list below.

- [x] The size of the returned array c is $m + n$.

Correct
Note that it is not possible to merge in general without extra storage or additional time complexity.

- [x] The merge algorithm runs in time $\Theta(m + n)$.

Correct
Correct.

- [x] If the loop in lines 14-16 execute for at least one iteration, then the loop in lines 17-19 will execute for zero iterations

Correct
Correct: if the while loop in lines 14-16 run, it means that we finished with array b and still have left over elements to merge but only from array a.

- [x] Whenever line 7 is reached during the execution of the algorithm, the array c is in fact the merge of the sub-arrays $a[0, \ldots, i-1]$ and $b[0, \ldots, j-1]$.

Correct
This is correct.

- [x] Whenever line 7 is reached during the execution of the algorithm and $j < m$, the element $b[j]$ is greater than or equal to every element of the array c.

Correct
Correct: this is an important fact that we need to guarantee that merge works correctly.

- [ ] Whenever line 7 is reached during the execution of the algorithm, $a[i] > b[j]$.

- [x] Whenever line 17 is reached during the execution, the array c is the merge of the entire array a and the subarray $b[0, ..., j-1]$.

Correct
Correct.

2. Consider the recursive implementation of mergesort algorithm:

3 / 3 points

```
1  def mergesort(a, left, right):
2      if right <= left:
```

```
3        return # Nothing to do -- base case
4    mid = (left + right)//2 # In python // is integer division (and round down to nearest integ
5    mergesort(a, left, mid) # recursively sort a from left..mid (inclusive)
6    mergesort(a, mid + 1 , right) # recursively sort a from mid+1... right
7    mergeAndCopyBack(a, left, mid, right ) # runs in time Theta(right - left + 1)
8    # mergeAndCopyBack will copy back the merge of
9    # a[left...mid] and a[mid+1...right] into a[left...right]
```

The size region to sort for a call with indices left, right is given by (right - left + 1).

Given an array a, we call `mergesort(a, 0, len(a) -1)`.

Select all the correct facts from the list below.

- [x] If $k = (right - left + 1)$ is the size of the region to be sorted, then the recursive calls in lines 5 and 6 will involve regions of size at most $(k + 1)/2$.

Correct
Correct.

- [x] If the size of the initial array to be sorted is 256 (2^8) then the size of the regions to sort recursive calls in lines 5/6 is 128.

Correct

- [x] if the size of the initial array to be sorted is 16 (2^4), then during the execution of the algorithm, there will be 4 calls to mergesort with regions of size 4.

Correct

- [ ] If the size of the initial array to be sorted is 16 (2^4), then during the execution of the algorithm, there will be 8 calls to mergesort with regions of size 1.