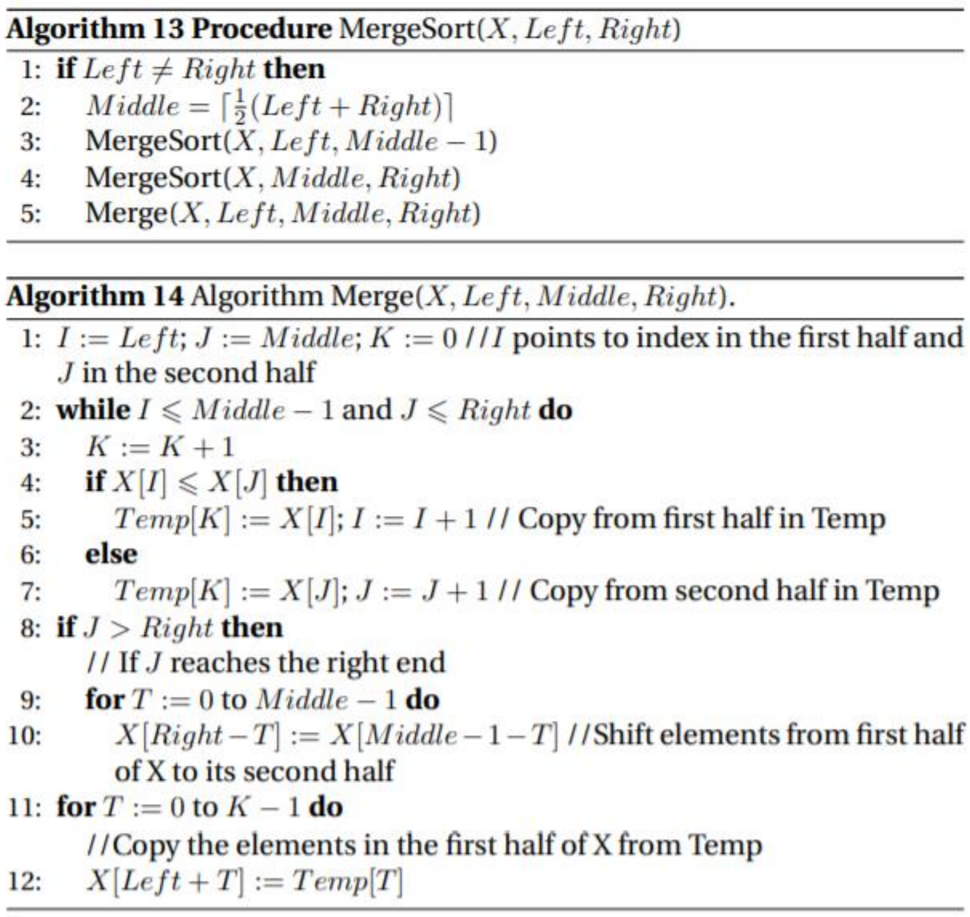
Dominic Grant

Programming Assignment 1 Explanation:

The Pseudocode is basically



Algorithm 13 Procedure MergeSort($X, Left, Right$)

```
1: if $Left \neq Right$ then
2:     $Middle = \lceil \frac{1}{2}(Left + Right) \rceil$
3:     MergeSort($X, Left, Middle - 1$)
4:     MergeSort($X, Middle, Right$)
5:     Merge($X, Left, Middle, Right$)
```

Algorithm 14 Algorithm Merge($X, Left, Middle, Right$).

```
1: $I := Left$; $J := Middle$; $K := 0$ //$I$ points to index in the first half and
   $J$ in the second half
2: while $I \leqslant Middle - 1$ and $J \leqslant Right$ do
3:     $K := K + 1$
4:     if $X[I] \leqslant X[J]$ then
5:         $Temp[K] := X[I]$; $I := I + 1$ // Copy from first half in Temp
6:     else
7:         $Temp[K] := X[J]$; $J := J + 1$ // Copy from second half in Temp
8: if $J > Right$ then
       // If $J$ reaches the right end
9:     for $T := 0$ to $Middle - 1$ do
10:        $X[Right - T] := X[Middle - 1 - T]$ //Shift elements from first half
           of X to its second half
11: for $T := 0$ to $K - 1$ do
       //Copy the elements in the first half of X from Temp
12:    $X[Left + T] := Temp[T]$
```

but with a few differences. First, I wait until the end of the while loops to add up, I have a bunch of checks I have in place in both Algorithm 13 and 14 to see if the two current numbers in use add up to the target, I have more conditions for both algorithms an example being I use a condition that switches off the target checking once it has been found, and I have a main() which basically just asks the user for the array length and integers, the target, and a couple lines of code that print out the sorted array at the very end. Overall nothing that mindblowing that isn't already seen in this original pseudocode from the book here.

Algorithm conditions here:

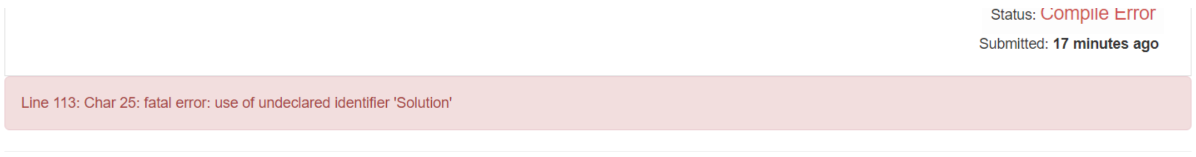
```
void mergeSort(int *array, int l, int h, int tar, int check)
```
the array, low, high, target, check

```
void merge(int *array, int l, int h, int m, int tar, int CHECK)
```
array, low, high, middle, target, check

In plain english, it works because the program separates the array into smaller sections which it goes through sorting, WHILE at the same time checking each and every instance to see if the 2 numbers add up to the given target. Once it finds the target, it stops checking and finishes up sorting.

It's correct because it's correct. It's the same as the pseudocode given in the book more or less with only a small if check that tries to find the target as the array is being sorted. And since it is a merge sort, we know that it has a complexity of nlogn. There isn't any additional loops that would cause the time to go over except perhaps the sort array print out at the very end of the code.

The code was not excepted my leetcode, but that is only due to the fact that I didn't use the "Solution" Class it was trying to force the user to use.



Status: Compile Error
Submitted: **17 minutes ago**

```
Line 113: Char 25: fatal error: use of undeclared identifier 'Solution'
```

Considering how my code is only 106 lines, it's obviously something with leetcode

https://leetcode.com/submissions/detail/355168266/