```
# Program #2 (Question #2)

## Name: Jermaine Presbery
## Date: 10-20-21
## Assignment #3


# --------------------- Merge Sort Function --------------------------- #
def mergesort(arr):
  if len(arr) > 1:
    middle = len(arr) // 2    # This will be the middle number within the array
    left = arr[:middle]
    right = arr[middle:]

    # Calling merge sort Function to divide the array's left and right side recursively
    mergesort(left)
    mergesort(right)




  # Setting index to zero for each list to the left most
    i = 0  # left side's leftmost index
    j = 0  # right side's leftmost index
    merged_array = 0  # This is the merged array index to store both left and right side of t

    # Merging of the individual lists
    while i < len(left) and j < len(right):
      if left[i] <= right[j]:
        arr[merged_array] = left[i]
        i = i + 1  # This moves to the next element within the left side of array
      else:
        arr[merged_array] = right[j]
        j = j + 1   # This moves to the next element within the right side of array
      merged_array += 1     # This will increase the length of the merged array in every whil


    # Obtaining remaining numbers
    while i < len(left):
      arr[merged_array] = left[i]
      i += 1
      merged_array += 1

    while j < len(right):
      arr[merged_array] = right[j]
      j += 1
      merged_array += 1

  #--------------------------------------------------------------------#
```

```
# -------------------- Binary Search Function ------------------------ #

# Parameter 1 (arr) = user-defined Array
# Parameter 2 (x) = desired non-negative integer within an array that the user wants to find

def binarysearch(arr, x):

    middle = len(arr) // 2  # Going with the divide and conquer approach to best compare the

    if x == arr[middle]:
      return x
    elif x > arr[middle]:
      return binarysearch(arr[middle + 1:], x) # This will go towards the upperBounds of the
    elif x < arr[middle]:
      return binarysearch(arr[:middle], x) # This will go towards the lowerBounds of the arra
    else:
      return -1 # Return -1 if x is not found within the array


# Importing random function to creating a random array
import random

# Setting a random seed so every time the program is ran the output of random numbers will al
random.seed(0)

# Declaring an array variable of length 100 with random non-negative integers
A = [random.randint(0, 100) for i in range(100)]


# Calling mergesort function to convert the unsorted array to a sorted array in increasing or
mergesort(A)

# Declaring an array variable that is indexed from 0:10
z = A[0:10]

# Printing the array from index 0:10
print("Array spanning from index 0 - 10: ")
print(z, end= '\n')


# Calling binarysearch function for 2 elements in an array
print("Calling binary search for 2 elements within the array: ")
print("Element within Array: ", binarysearch(z, 5))
print("Element within Array: ", binarysearch(z, 8))

# Calling binearysearch function for 1 element outside of the bounds of the array
binarysearch(z, 3)
```