NAME: MD. NAYEEM MOLLA

Subject: Algorithm Design and Analysis

Experiment II: Divide and Conquer

1. Purpose

Apply the divide and conquer algorithm to solve the practical problems and implement it in programming language.

2. Main requirements

(1) Write and debug the recursive program of binary search.

(2) Write and debug the recursive program of merge sort.

3. Instrument and equipment

PC-compatible (language-free).

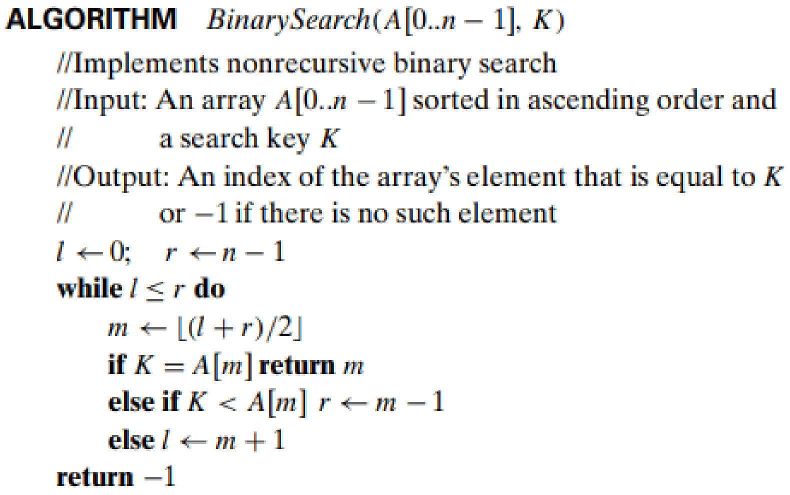
4. Algorithm's principles

4.1 Binary Search

✧ Algorithmic strategy (non-descending): Binary search is a remarkably efficient algorithm for searching in a sorted array. It works by comparing a search key K with the array’s middle element A[m]. If they match, the algorithm stops; otherwise, the same operation is repeated recursively for the first half of the array if K < A[m], and for the second half if K > A[m]:

$$\begin{array}{c} K \\ \updownarrow \\ \underbrace{A[0]\ldots A[m-1]}_{\substack{\text{search here if}\\ K<A[m]}}\ \ A[m]\ \ \underbrace{A[m+1]\ldots A[n-1]}_{\substack{\text{search here if}\\ K>A[m]}}. \end{array}$$

- **Pseudocode**



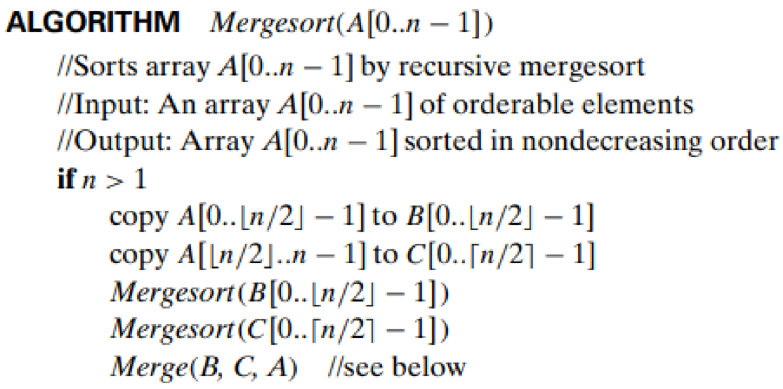
```
ALGORITHM  BinarySearch(A[0..n − 1], K)
    //Implements nonrecursive binary search
    //Input: An array A[0..n − 1] sorted in ascending order and
    //         a search key K
    //Output: An index of the array’s element that is equal to K
    //         or −1 if there is no such element
    l ← 0;   r ← n − 1
    while l ≤ r do
        m ← ⌊(l + r)/2⌋
        if K = A[m] return m
        else if K < A[m] r ← m − 1
        else l ← m + 1
    return −1
```

4.2 Merge Sort

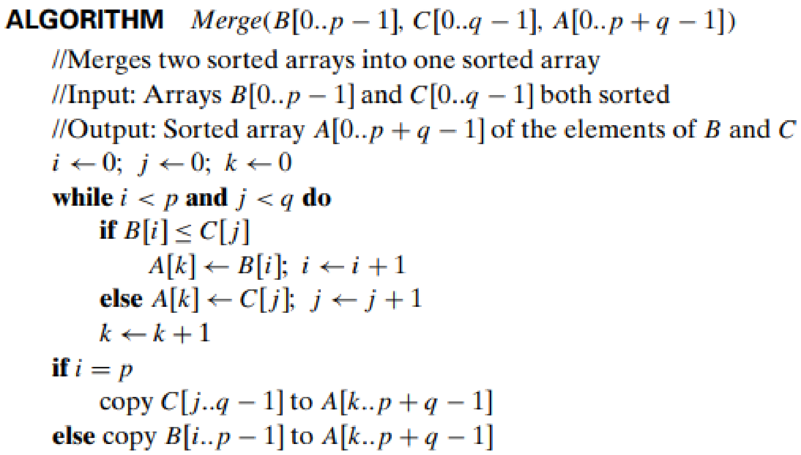
- Algorithmic strategy (non-descending): Mergesort is a perfect example of a successful application of the divide-and conquer technique. It sorts a given array A[0..n − 1] by dividing it into two halves A[0..$\lfloor n/2 \rfloor$ − 1] and A[$\lfloor n/2 \rfloor$.. n − 1], sorting each of them recursively, and then merging the two smaller sorted arrays into a single sorted one.

  The merging of two sorted arrays can be done as follows. Two pointers (array indices) are initialized to point to the first elements of the arrays being merged. The elements pointed to are compared, and the smaller of them is added to a new array being constructed; after that, the index of the smaller element is incremented to point to its immediate successor in the array it was copied from. This operation is repeated until one of the two given arrays is exhausted, and then the remaining elements of the other array are copied to the end of the new array.

- **Pseudocode**



```
ALGORITHM  Mergesort(A[0..n − 1])
    //Sorts array A[0..n − 1] by recursive mergesort
    //Input: An array A[0..n − 1] of orderable elements
    //Output: Array A[0..n − 1] sorted in nondecreasing order
    if n > 1
        copy A[0..⌊n/2⌋ − 1] to B[0..⌊n/2⌋ − 1]
        copy A[⌊n/2⌋..n − 1] to C[0..⌈n/2⌉ − 1]
        Mergesort(B[0..⌊n/2⌋ − 1])
        Mergesort(C[0..⌈n/2⌉ − 1])
        Merge(B, C, A)    //see below
```



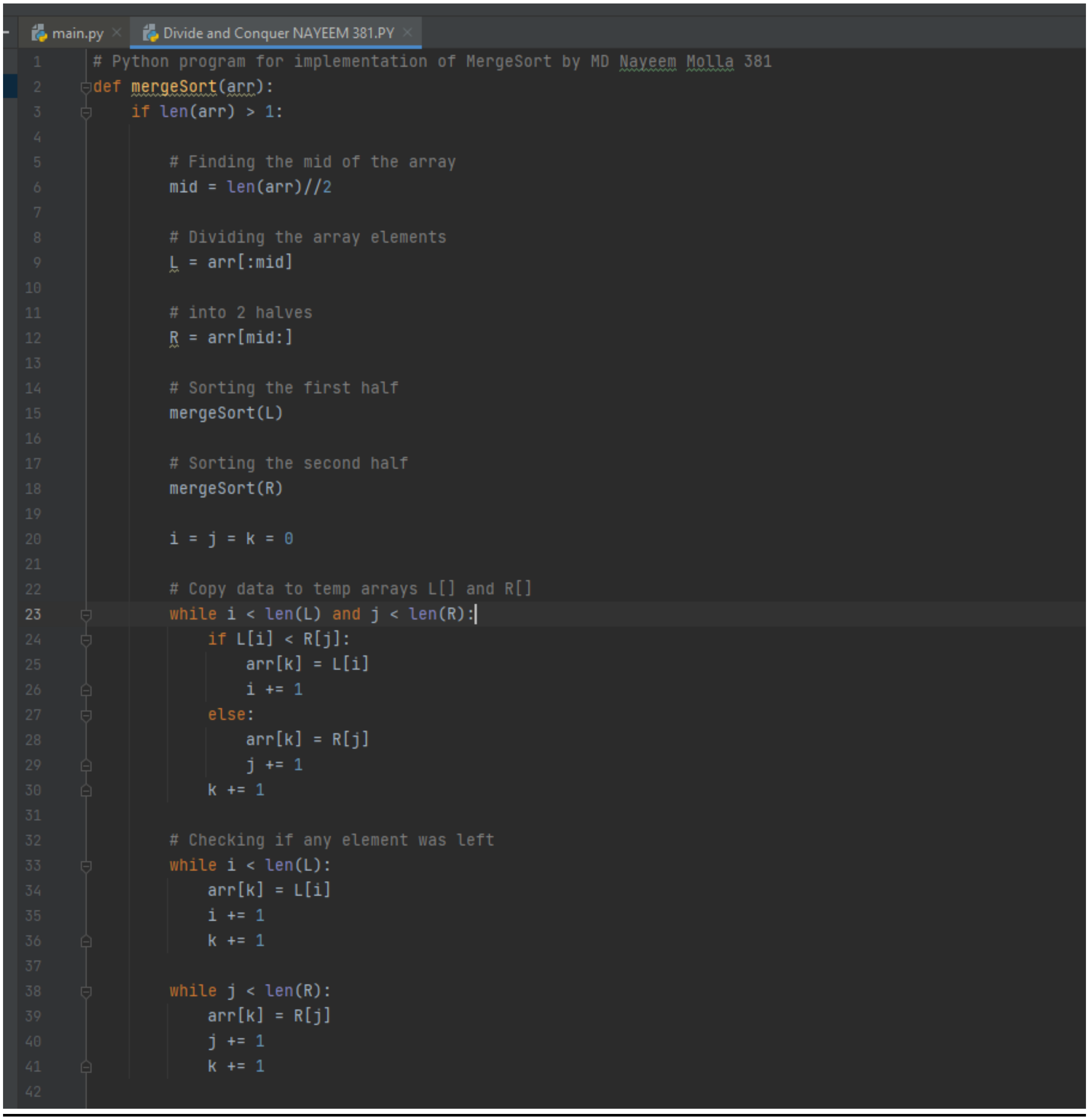
```
ALGORITHM  Merge(B[0..p − 1], C[0..q − 1], A[0..p + q − 1])
    //Merges two sorted arrays into one sorted array
    //Input: Arrays B[0..p − 1] and C[0..q − 1] both sorted
    //Output: Sorted array A[0..p + q − 1] of the elements of B and C
    i ← 0; j ← 0; k ← 0
    while i < p and j < q do
        if B[i] ≤ C[j]
            A[k] ← B[i]; i ← i + 1
        else A[k] ← C[j]; j ← j + 1
        k ← k + 1
    if i = p
        copy C[j..q − 1] to A[k..p + q − 1]
    else copy B[i..p − 1] to A[k..p + q − 1]
```

Source code:

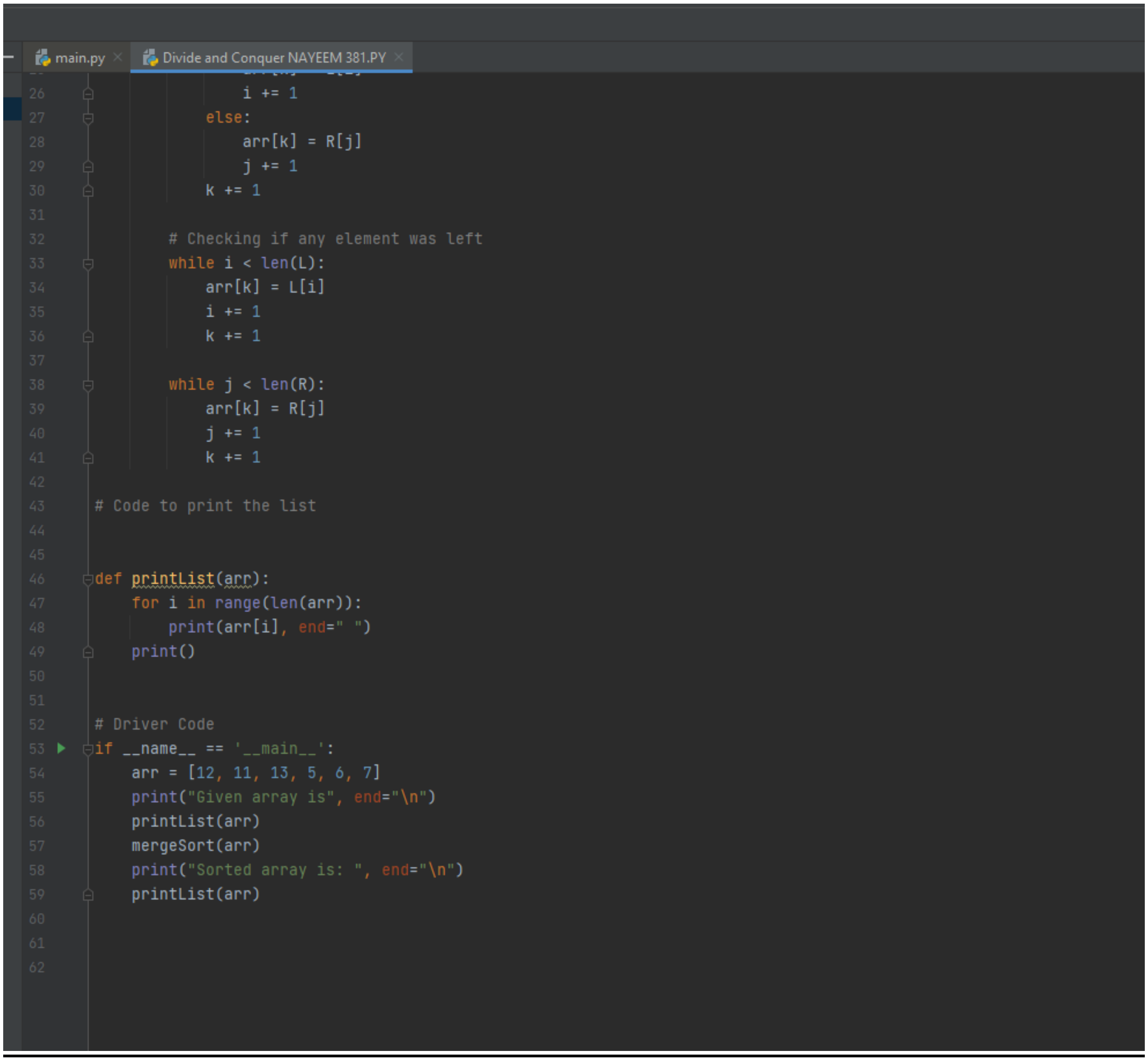
```
    # Python program for implementation of MergeSort by MD Nayeem Molla 381
def mergeSort(arr):
  if len(arr) > 1:


    # Finding the mid of the array
    mid = len(arr)//2


    # Dividing the array elements
    L = arr[:mid]


    # into 2 halves
    R = arr[mid:]

    # Sorting the first half
    mergeSort(L)

    # Sorting the second half
    mergeSort(R)

    i = j = k = 0

    # Copy data to temp arrays L[] and R[]
    while i < len(L) and j < len(R):
      if L[i] < R[j]:
        arr[k] = L[i]
        i += 1
      else:
        arr[k] = R[j]
        j += 1
      k += 1

    # Checking if any element was left
    while i < len(L):
      arr[k] = L[i]
      i += 1
      k += 1

    while j < len(R):
      arr[k] = R[j]
      j += 1
      k += 1

# Code to print the list
```

```
def printList(arr):
    for i in range(len(arr)):
        print(arr[i], end=" ")
    print()


# Driver Code
if __name__ == '__main__':
    arr = [12, 11, 13, 5, 6, 7]
    print("Given array is", end="\n")
    printList(arr)
    mergeSort(arr)
    print("Sorted array is: ", end="\n")
    printList(arr)
```

INPUT:

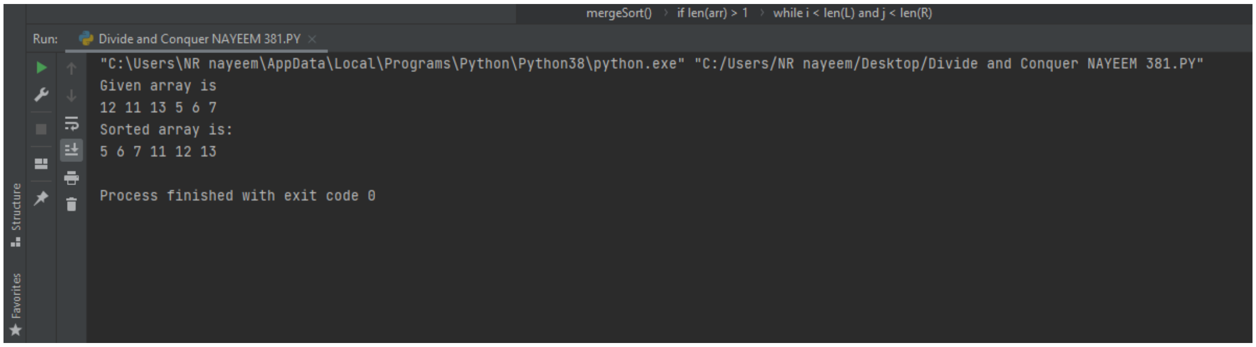


```python
# Python program for implementation of MergeSort by MD Nayeem Molla 381
def mergeSort(arr):
    if len(arr) > 1:

        # Finding the mid of the array
        mid = len(arr)//2

        # Dividing the array elements
        L = arr[:mid]

        # into 2 halves
        R = arr[mid:]

        # Sorting the first half
        mergeSort(L)

        # Sorting the second half
        mergeSort(R)

        i = j = k = 0

        # Copy data to temp arrays L[] and R[]
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1

        # Checking if any element was left
        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1

        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1

```



```python
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1

        # Checking if any element was left
        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1

        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1

# Code to print the list


def printList(arr):
    for i in range(len(arr)):
        print(arr[i], end=" ")
    print()


# Driver Code
if __name__ == '__main__':
    arr = [12, 11, 13, 5, 6, 7]
    print("Given array is", end="\n")
    printList(arr)
    mergeSort(arr)
    print("Sorted array is: ", end="\n")
    printList(arr)
```

OUTPUT:


mergeSort() › if len(arr) > 1 › while i < len(L) and j < len(R)
Run:
Divide and Conquer NAYEEM 381.PY
Structure
Favorites

```
"C:\Users\NR nayeem\AppData\Local\Programs\Python\Python38\python.exe" "C:/Users/NR nayeem/Desktop/Divide and Conquer NAYEEM 381.PY"
Given array is
12 11 13 5 6 7
Sorted array is:
5 6 7 11 12 13

Process finished with exit code 0
```