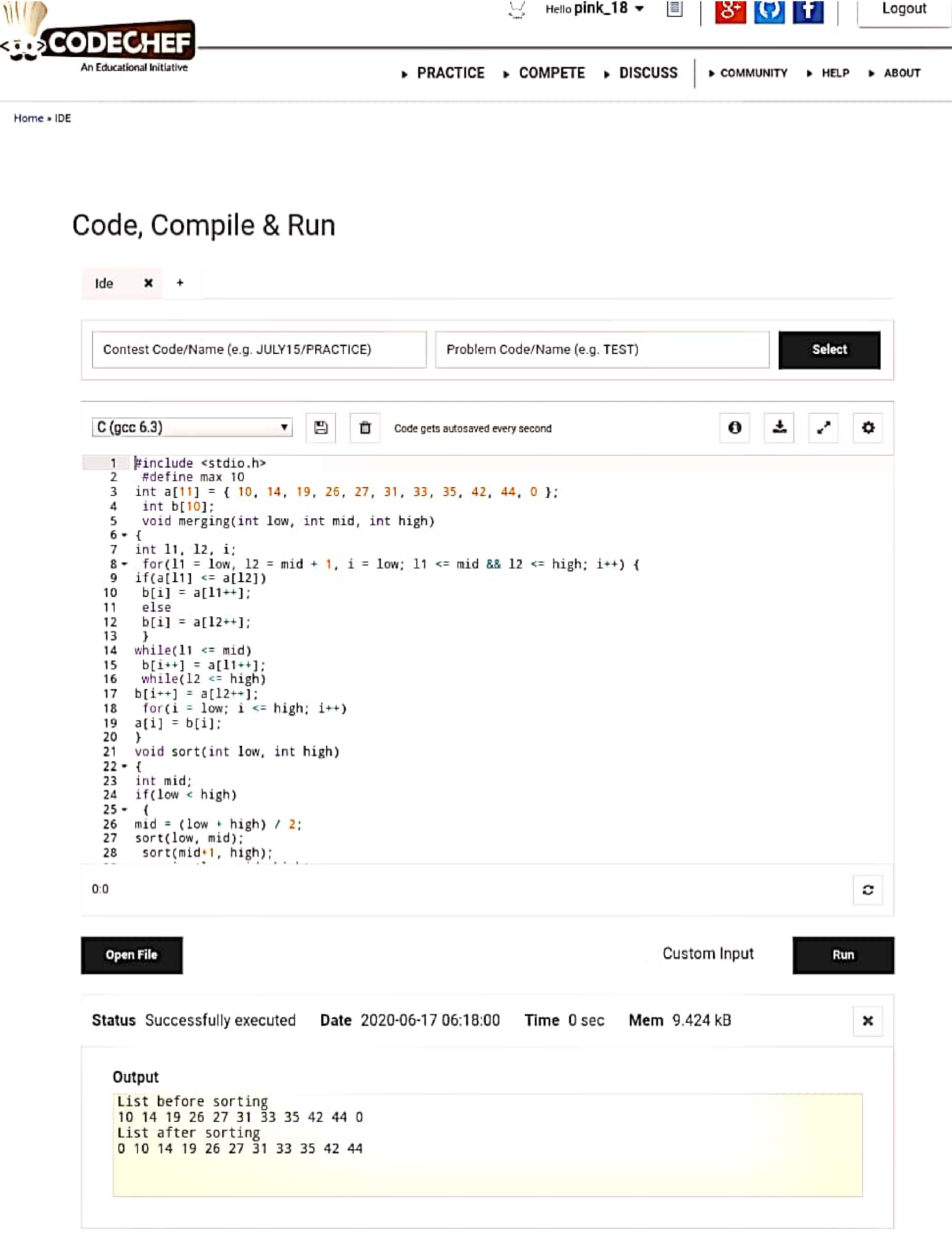


CODECHEF
An Educational Initiative
Hello pink_18
Logout
PRACTICE
COMPETE
DISCUSS
COMMUNITY
HELP
ABOUT
Home » IDE

Code, Compile & Run

Ide
Contest Code/Name (e.g. JULY15/PRACTICE)
Problem Code/Name (e.g. TEST)
Select
C (gcc 6.3)
Code gets autosaved every second

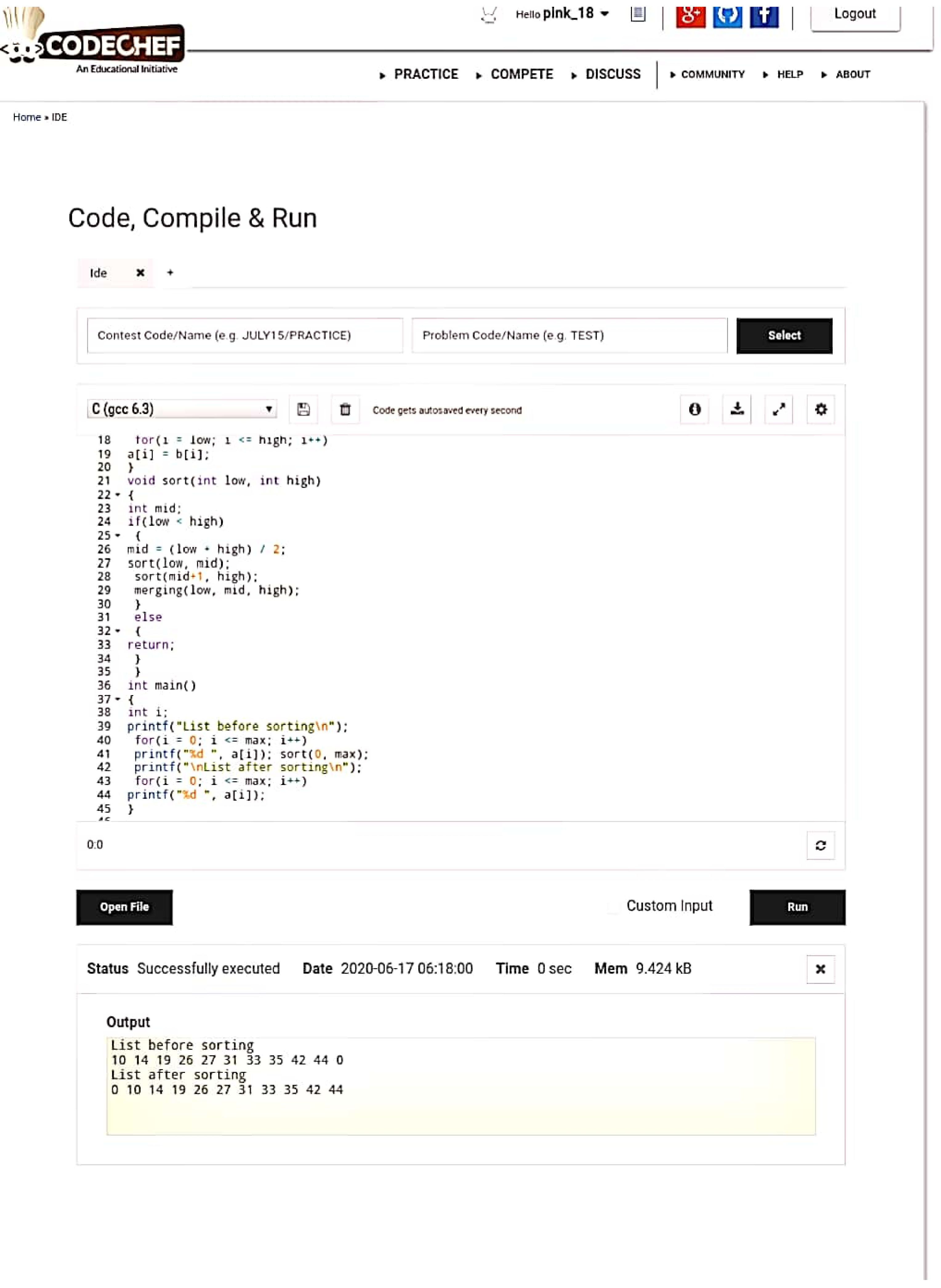
```c
#include <stdio.h>
 #define max 10
int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };
 int b[10];
 void merging(int low, int mid, int high)
{
int l1, l2, i;
 for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
if(a[l1] <= a[l2])
 b[i] = a[l1++];
 else
 b[i] = a[l2++];
 }
while(l1 <= mid)
 b[i++] = a[l1++];
 while(l2 <= high)
b[i++] = a[l2++];
 for(i = low; i <= high; i++)
a[i] = b[i];
}
void sort(int low, int high)
{
int mid;
if(low < high)
 {
mid = (low + high) / 2;
sort(low, mid);
 sort(mid+1, high);
```

0:0
Open File
Custom Input
Run

Status Successfully executed **Date** 2020-06-17 06:18:00 **Time** 0 sec **Mem** 9.424 kB

Output

```
List before sorting
10 14 19 26 27 31 33 35 42 44 0
List after sorting
0 10 14 19 26 27 31 33 35 42 44
```


Hello pink_18
Logout
CODECHEF
An Educational Initiative
PRACTICE
COMPETE
DISCUSS
COMMUNITY
HELP
ABOUT
Home » IDE

Code, Compile & Run

Ide
Contest Code/Name (e.g. JULY15/PRACTICE)
Problem Code/Name (e.g. TEST)
Select
C (gcc 6.3)
Code gets autosaved every second

```
 for(i = low; i <= high; i++)
a[i] = b[i];
}
void sort(int low, int high)
{
int mid;
if(low < high)
 {
mid = (low + high) / 2;
sort(low, mid);
 sort(mid+1, high);
 merging(low, mid, high);
 }
 else
 {
return;
 }
 }
int main()
{
int i;
printf("List before sorting\n");
 for(i = 0; i <= max; i++)
 printf("%d ", a[i]); sort(0, max);
 printf("\nList after sorting\n");
 for(i = 0; i <= max; i++)
printf("%d ", a[i]);
}
```

0:0
Open File
Custom Input
Run

Status Successfully executed Date 2020-06-17 06:18:00 Time 0 sec Mem 9.424 kB

Output

```
List before sorting
10 14 19 26 27 31 33 35 42 44 0
List after sorting
0 10 14 19 26 27 31 33 35 42 44
```

Priyanka shet
4AL19CS070

Algorithm : (merge sort)

step1 : start

step2 : Merge sort(arr[], l, r) where l is the index of the first element & r is the index of the last element.

if r > l

Step 3 : Find the middle index of the array to divide it in two halves.

$m = (l+r)/2$

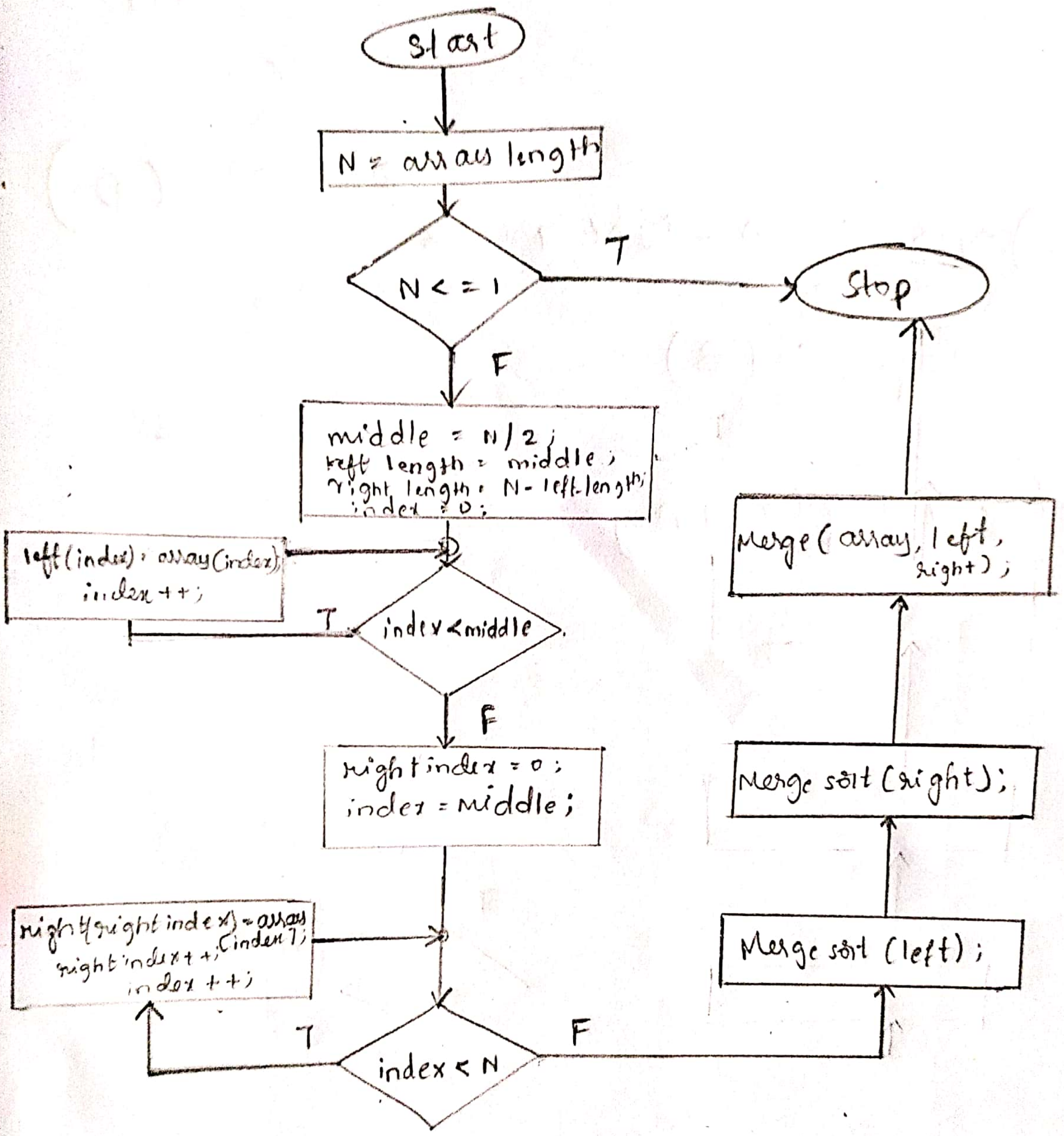
step 4 : Call Mergesort for first half :
merge sort (array, l, m)

step 5 : call merge sort for second half :
merge sort (array, m+1, r)

step 6 : Recursively, merge the two halves in a sorted manner, so that only one sorted array is left :
merge (array, l, m, r)

step 7 : stop

(ii) Flowchart :



Start
N = array length
N <= 1
T
Stop
F
middle = N/2;
left length = middle;
right length = N - left length;
index = 0;
left(index) = array(index);
index ++;
T
index < middle
F
right index = 0;
index = middle;
right(right index) = array (index);
right index ++;
index ++;
T
index < N
F
Merge sort (left);
Merge sort (right);
Merge (array, left, right);