Green University of Bangladesh

Department of Computer Science and Engineering(CSE)

Faculty of Sciences and Engineering

Semester: (Summer , Year:2022), B.Sc. in CSE (Day)

LAB REPORT NO : 3

Course Title: Structured Programming Lab

Course Code: CSE 106 Section:PC-213DA

Lab Experiment Name: Bubble Sort , Quick Sort , Merge Sort Using Array.

Student Details

Name	ID
MD Dulal Hossain	213902116

Lab Date	**:**	**06-07-2022**
Submission Date	**:**	**20-07-2022**
Course Teacher's Name	**:**	**Farhana Akther Sunny**

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: …………………………………	**Signature:.....................**
Comments:...	**Date:..............................**

//Quick Sort

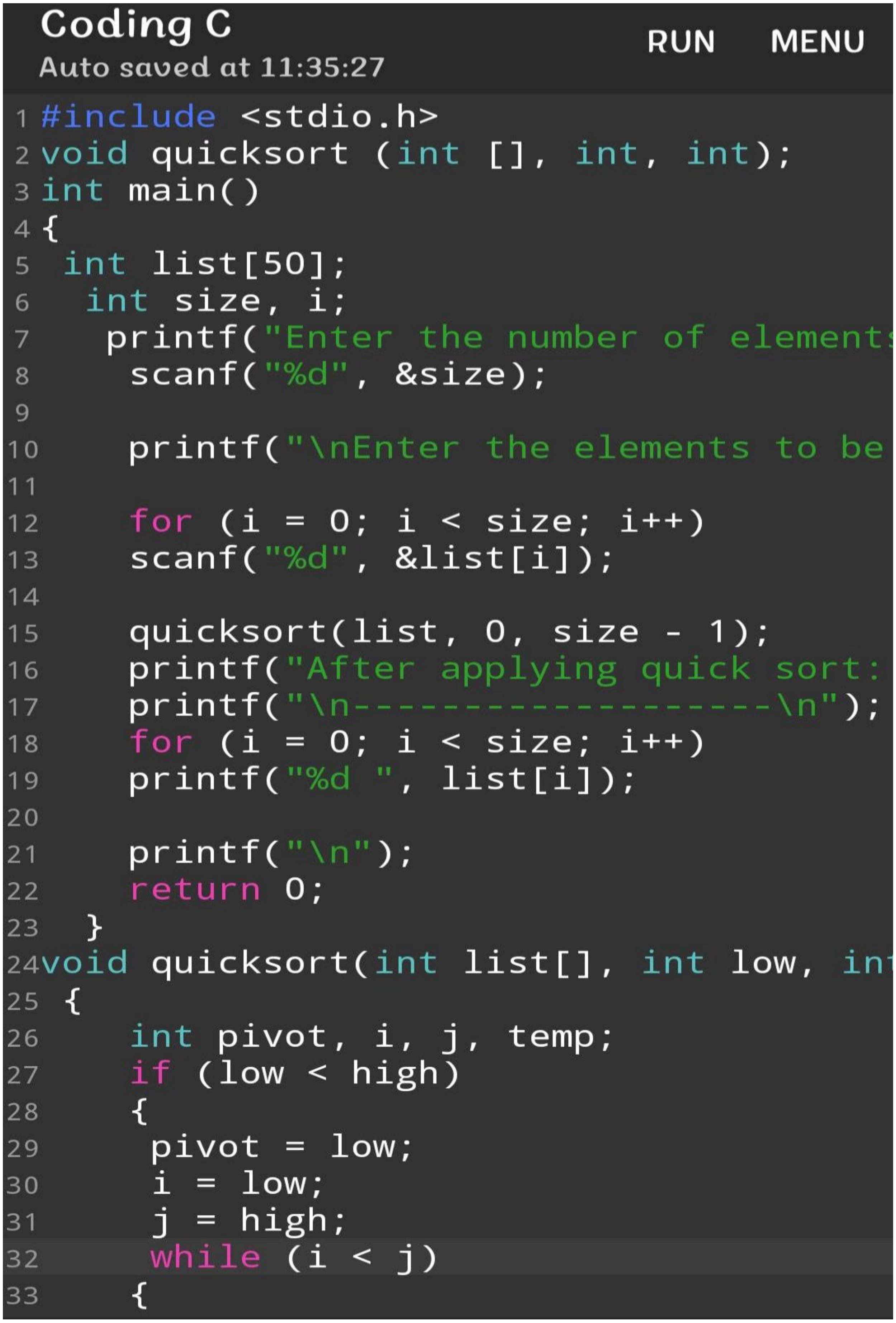
Algorithm :

Step 1. Declare array size.
Step 2. User input array.
Step 3. Using structure to quicksort.
Step 4. Find the low and high elements then fix you let low = beg and compare the number if beg getter then   compare number , beg go to last and compare number is new beg.
Step 5. This condition Continue until beg== compare number.
Step 6. Print the sorted array.
Step 7. End

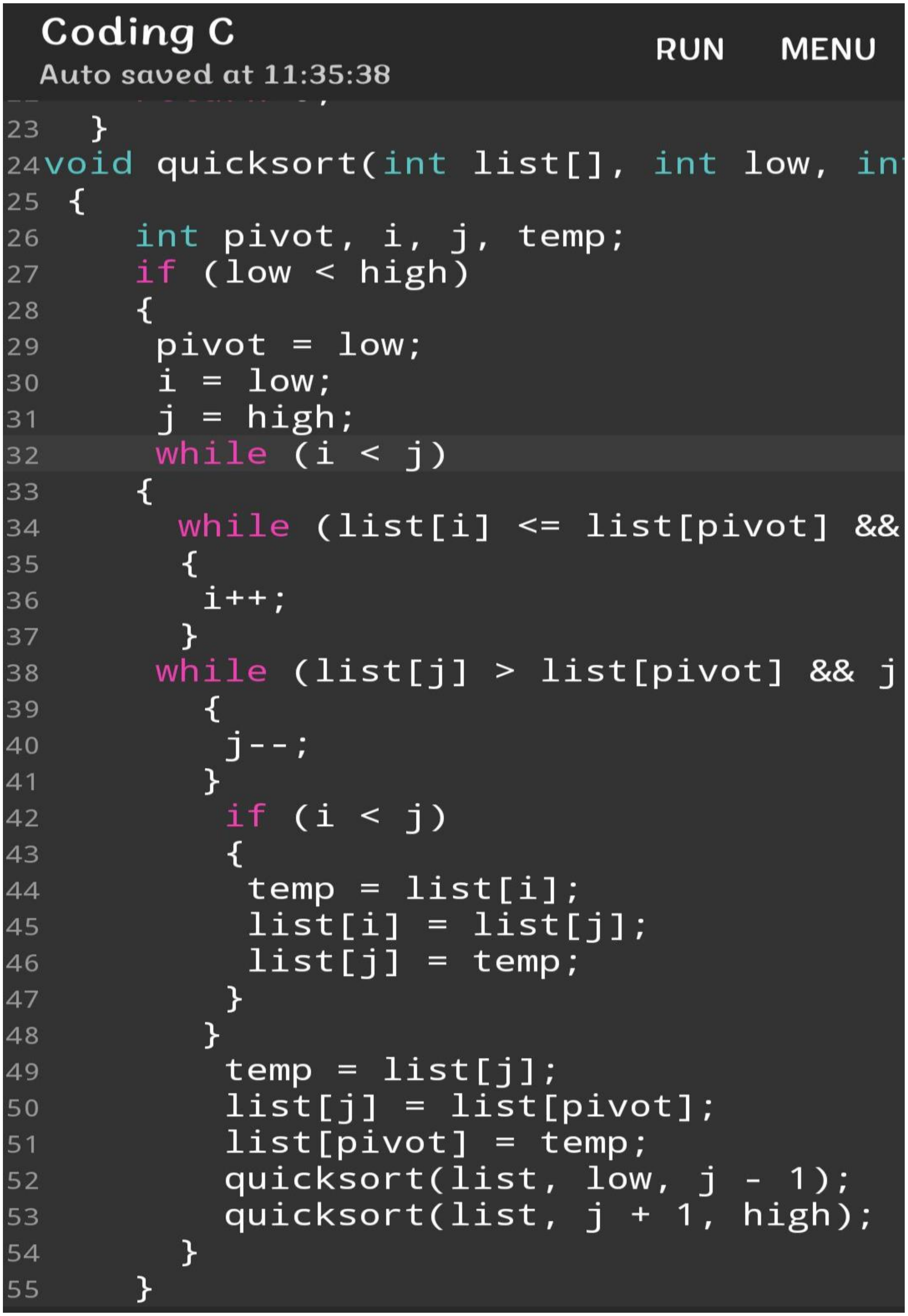
Source Code :

```
#include <stdio.h>
Void quicksort (int [], int, int);
Int main()
{
 Int list[50];
  Int size, I;
   Printf("Enter the number of elements: ");
    Scanf("%d", &size);

    Printf("\nEnter the elements to be sorted: ");

    For (I = 0; I < size; i++)
    Scanf("%d", &list[i]);

    Quicksort(list, 0, size – 1);
    Printf("After applying quick sort: \n");
    Printf("\n-------------------\n");
    For (I = 0; I < size; i++)
    Printf("%d ", list[i]);

    Printf("\n");
    Return 0;
  }
Void quicksort(int list[], int low, int high)
 {
    Int pivot, I, j, temp;
    If (low < high)
    {
     Pivot = low;
     I = low;
     J = high;
     While (I < j)
    {
      While (list[i] <= list[pivot] && I <= high)
      {
       I++;
      }
     While (list[j] > list[pivot] && j >= low)
       {
       j--;
       }
       If (I < j)
       {
        Temp = list[i];
        List[i] = list[j];
        List[j] = temp;
       }
      }
       Temp = list[j];
       List[j] = list[pivot];
       List[pivot] = temp;
       Quicksort(list, low, j – 1);
       Quicksort(list, j + 1, high);
     }
    }
```

My Code :



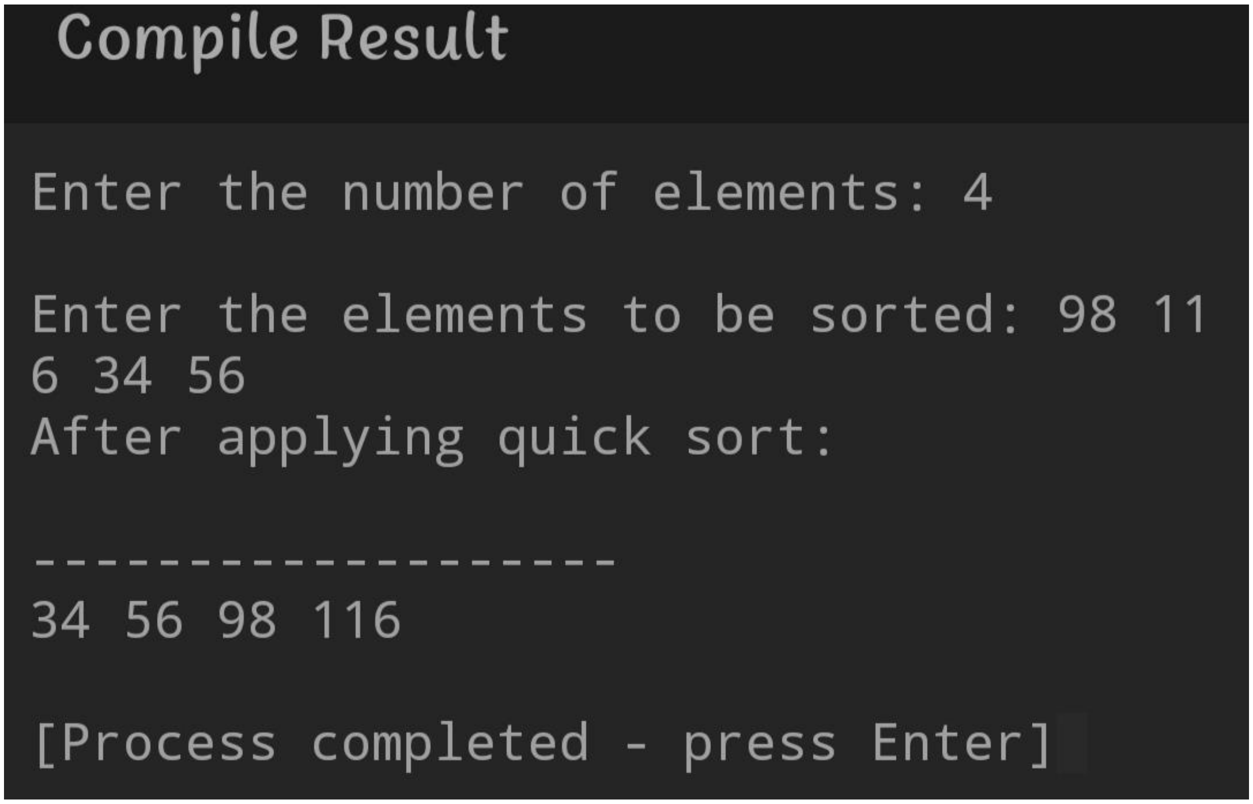
```c
#include <stdio.h>
void quicksort (int [], int, int);
int main()
{
 int list[50];
  int size, i;
   printf("Enter the number of element
    scanf("%d", &size);

    printf("\nEnter the elements to be

    for (i = 0; i < size; i++)
    scanf("%d", &list[i]);

    quicksort(list, 0, size - 1);
    printf("After applying quick sort:
    printf("\n--------------------\n");
    for (i = 0; i < size; i++)
    printf("%d ", list[i]);

    printf("\n");
    return 0;
  }
void quicksort(int list[], int low, in
 {
    int pivot, i, j, temp;
    if (low < high)
    {
     pivot = low;
     i = low;
     j = high;
     while (i < j)
    {
```



Coding C
Auto saved at 11:35:38
RUN MENU

```
  }
void quicksort(int list[], int low, in
 {
    int pivot, i, j, temp;
    if (low < high)
    {
     pivot = low;
     i = low;
     j = high;
     while (i < j)
    {
      while (list[i] <= list[pivot] &&
      {
       i++;
      }
     while (list[j] > list[pivot] && j
       {
        j--;
       }
        if (i < j)
        {
         temp = list[i];
         list[i] = list[j];
         list[j] = temp;
        }
       }
        temp = list[j];
        list[j] = list[pivot];
        list[pivot] = temp;
        quicksort(list, low, j - 1);
        quicksort(list, j + 1, high);
      }
    }
```

Out put :



```
Enter the number of elements: 4

Enter the elements to be sorted: 98 11
6 34 56
After applying quick sort:

--------------------
34 56 98 116

[Process completed - press Enter]
```

//Merge Sort

Algorithm :

Step 1. Declare array size.
Step 2. User input array.
Step 3. Using structure to mergesort.

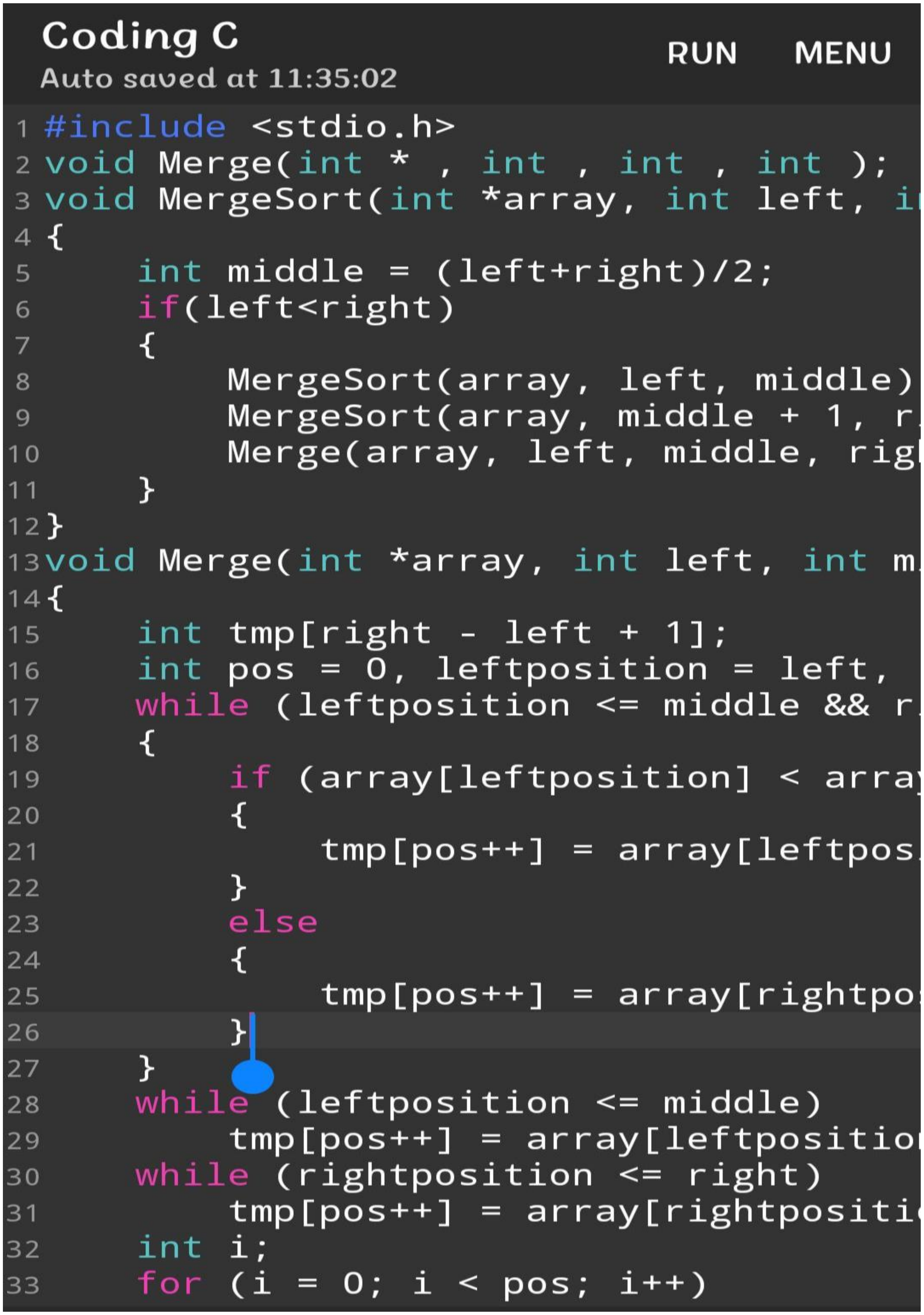
Step 4. Find the middle and let middle is beg and fix this, now we got two side name is left side & right side.
Step 5. Now left side & right side and also two Middle and again two middle let beg and compare.
Step 6. This conditions are continue at last.
Step 7. Print the sorted array.
Step 8. End

Source Code :

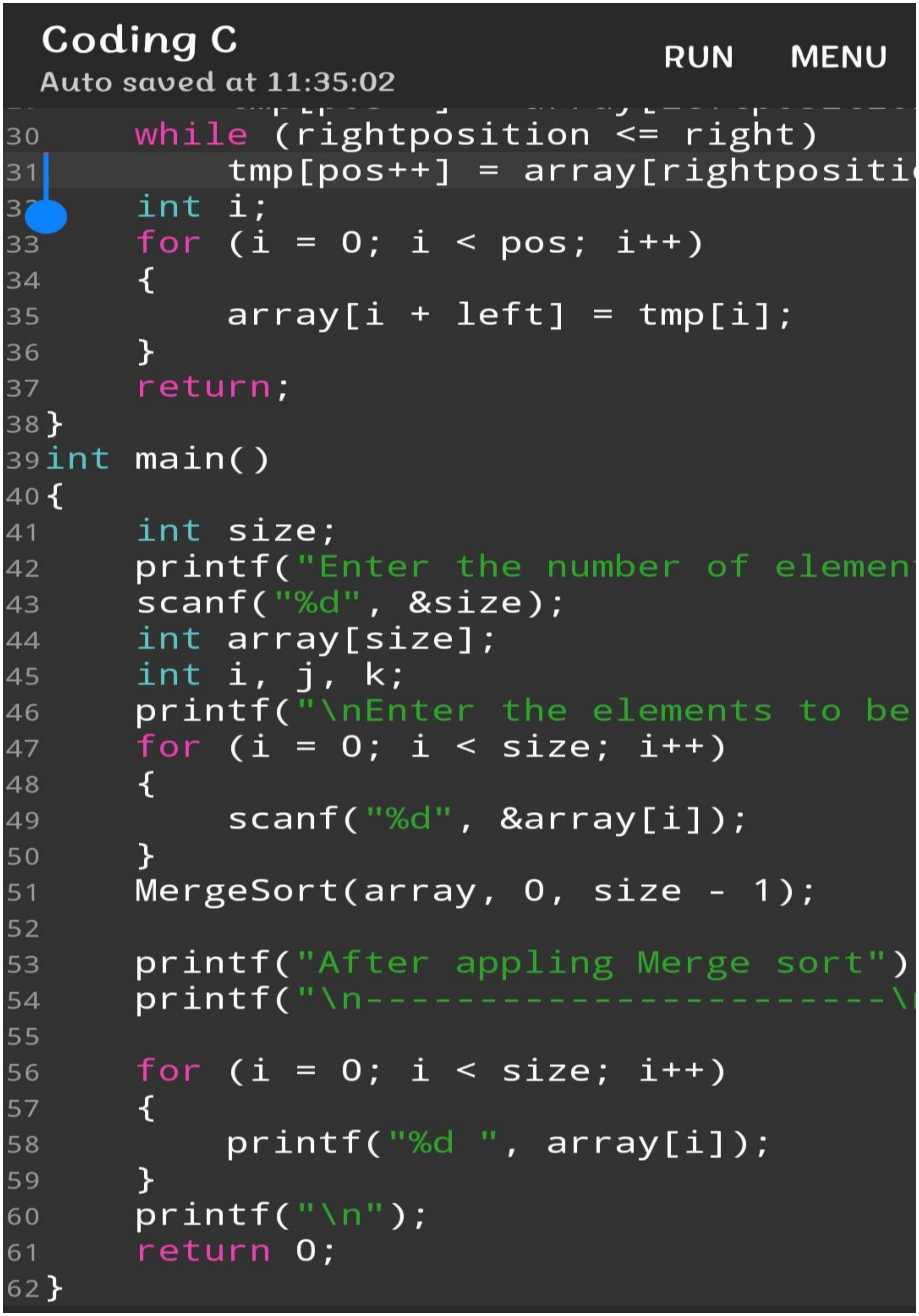
```
#include <stdio.h>
Void Merge(int * , int , int , int );
Void MergeSort(int *array, int left, int right)
{
   Int middle = (left+right)/2;
   If(left<right)
   {
      MergeSort(array, left, middle);
      MergeSort(array, middle + 1, right);
      Merge(array, left, middle, right);
   }
}
Void Merge(int *array, int left, int middle, int right)
{
   Int tmp[right – left + 1];
   Int pos = 0, leftposition = left, rightposition = middle + 1;
   While (leftposition <= middle && rightposition <= right)
   {
      If (array[leftposition] < array[rightposition])
      {
         Tmp[pos++] = array[leftposition++];
      }
      Else
      {
         Tmp[pos++] = array[rightposition++];
      }
```

```
    }
    While (leftposition <= middle)
       Tmp[pos++] = array[leftposition++];
    While (rightposition <= right)
       Tmp[pos++] = array[rightposition++];
    Int I;
    For (I = 0; I < pos; i++)
    {
       Array[I + left] = tmp[i];
    }
    Return;
}
Int main()
{
    Int size;
    Printf("Enter the number of elements : ");
    Scanf("%d", &size);
    Int array[size];
    Int I, j, k;
    Printf("\nEnter the elements to be sorted : ");
    For (I = 0; I < size; i++)
    {
       Scanf("%d", &array[i]);
    }
    MergeSort(array, 0, size – 1);

    Printf("After appling Merge sort");

    Printf("\n-----------------------\n");


    For (I = 0; I < size; i++)
    {
       Printf("%d ", array[i]);
    }
    Printf("\n");
    Return 0;
}
```

My Code :



```
#include <stdio.h>
void Merge(int * , int , int , int );
void MergeSort(int *array, int left, i
{
    int middle = (left+right)/2;
    if(left<right)
    {
        MergeSort(array, left, middle)
        MergeSort(array, middle + 1, r
        Merge(array, left, middle, rig
    }
}
void Merge(int *array, int left, int m
{
    int tmp[right - left + 1];
    int pos = 0, leftposition = left,
    while (leftposition <= middle && r
    {
        if (array[leftposition] < arra
        {
            tmp[pos++] = array[leftpos
        }
        else
        {
            tmp[pos++] = array[rightpo
        }
    }
    while (leftposition <= middle)
        tmp[pos++] = array[leftpositio
    while (rightposition <= right)
        tmp[pos++] = array[rightpositi
    int i;
    for (i = 0; i < pos; i++)
```



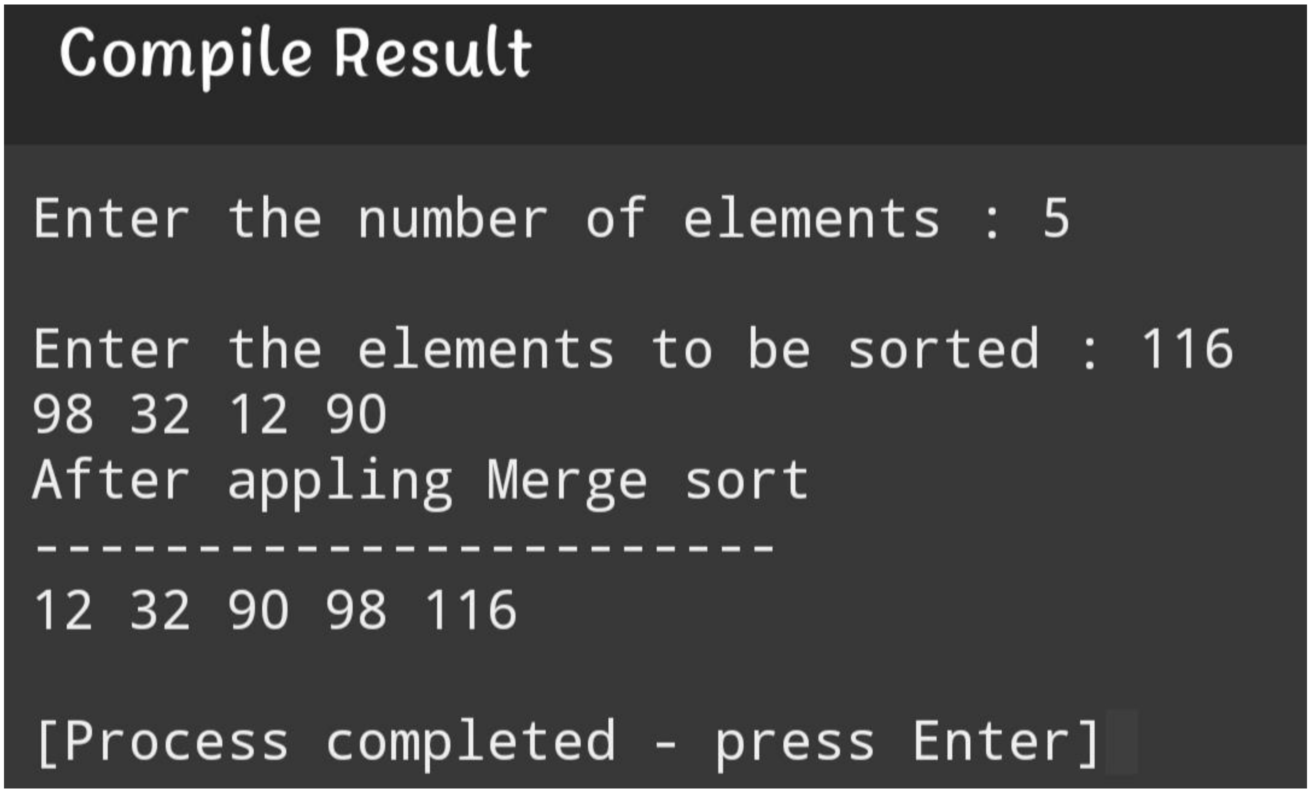
Coding C

Auto saved at 11:35:02

RUN MENU

```
    while (rightposition <= right)
        tmp[pos++] = array[rightpositi
    int i;
    for (i = 0; i < pos; i++)
    {
        array[i + left] = tmp[i];
    }
    return;
}
int main()
{
    int size;
    printf("Enter the number of elemen
    scanf("%d", &size);
    int array[size];
    int i, j, k;
    printf("\nEnter the elements to be
    for (i = 0; i < size; i++)
    {
        scanf("%d", &array[i]);
    }
    MergeSort(array, 0, size - 1);

    printf("After appling Merge sort")
    printf("\n------------------------\

    for (i = 0; i < size; i++)
    {
        printf("%d ", array[i]);
    }
    printf("\n");
    return 0;
}
```

Out put :



```
Enter the number of elements : 5

Enter the elements to be sorted : 116
98 32 12 90
After appling Merge sort
-----------------------
12 32 90 98 116

[Process completed - press Enter]
```