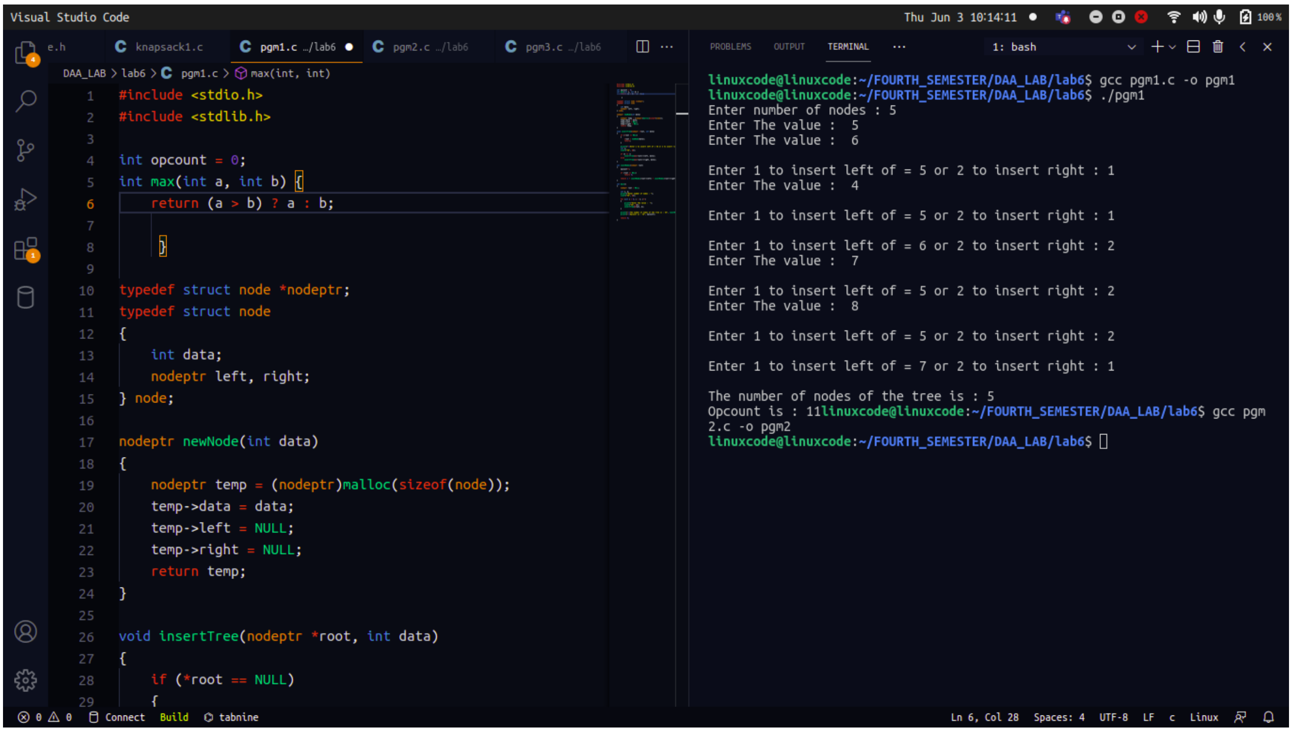
190905514 C SECTION 4th SEMESTER MOHAMMAD TOFIK

LAB EXCERCISE :

1) Find total number of nodes in a binary tree and analyze its efficiency. Obtain the experimental result of order of growth and plot the result.

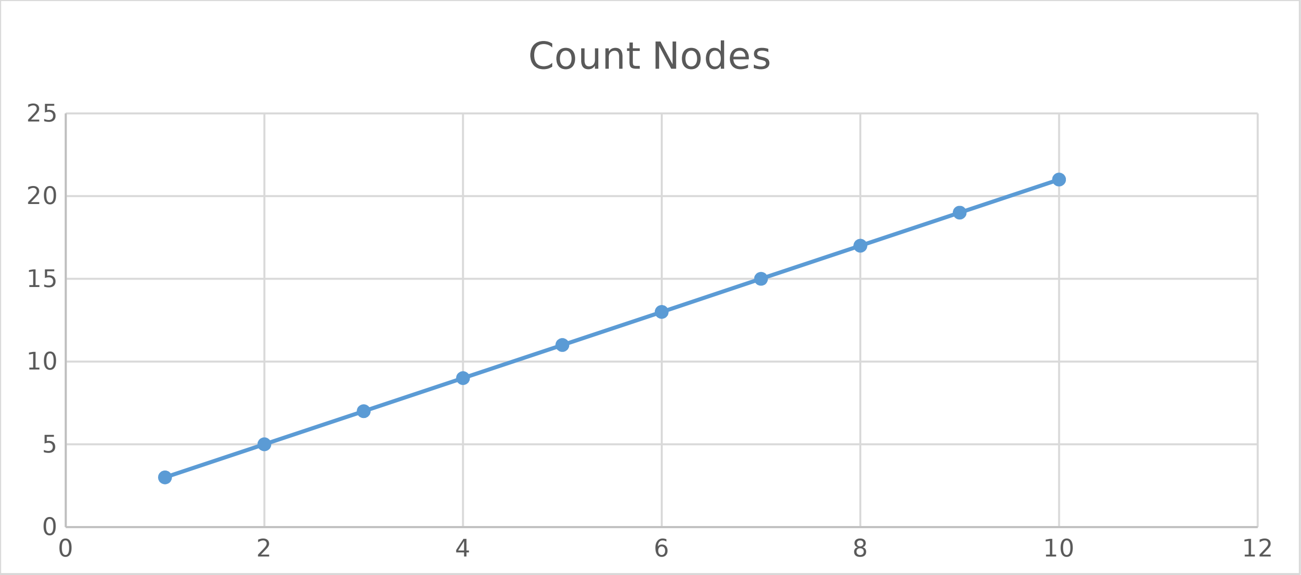
```
#include <stdio.h>
#include <stdlib.h>

int opcount = 0;
int max(int a, int b) {
return (a > b) ? a : b;

}

typedef struct node *nodeptr;
typedef struct node
{
int data;
nodeptr left, right;
} node;

nodeptr newNode(int data)
{
nodeptr temp = (nodeptr)malloc(sizeof(node));
temp->data = data;
temp->left = NULL;
temp->right = NULL;
return temp;
}

void insertTree(nodeptr *root, int data)
{
if (*root == NULL)
{
*root = newNode(data);
return;
}

printf("\nEnter 1 to insert left of = %d or 2 to insert right : ", (*root)-
>data);
int n;
scanf("%d", &n);

if (n == 1)
insertTree(&(*root)->left, data);
else
insertTree(&(*root)->right, data);
}

int countNodes(nodeptr root)
{
opcount++;

if (root == NULL)
return 0;

return 1 + countNodes(root->left) + countNodes(root->right);
}

int main()
{
nodeptr root = NULL;

int n, m;
printf("Enter number of nodes : ");
scanf("%d", &n);

for (int i = 0; i < n; i++)
{
printf("Enter The value : ");
scanf("%d", &m);
insertTree(&root, m);
}

printf("\nThe number of nodes of the tree is : %d", countNodes(root));
printf("\nOpcount is : %d", opcount);

return 0;
}
```

OUTPUT :



```c
#include <stdio.h>
#include <stdlib.h>

int opcount = 0;
int max(int a, int b) {
    return (a > b) ? a : b;

    }

typedef struct node *nodeptr;
typedef struct node
{
    int data;
    nodeptr left, right;
} node;

nodeptr newNode(int data)
{
    nodeptr temp = (nodeptr)malloc(sizeof(node));
    temp->data = data;
    temp->left = NULL;
    temp->right = NULL;
    return temp;
}

void insertTree(nodeptr *root, int data)
{
    if (*root == NULL)
    {
```

```
linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$ gcc pgm1.c -o pgm1
linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$ ./pgm1
Enter number of nodes : 5
Enter The value :  5
Enter The value :  6

Enter 1 to insert left of = 5 or 2 to insert right : 1
Enter The value :  4

Enter 1 to insert left of = 5 or 2 to insert right : 1

Enter 1 to insert left of = 6 or 2 to insert right : 2
Enter The value :  7

Enter 1 to insert left of = 5 or 2 to insert right : 2
Enter The value :  8

Enter 1 to insert left of = 5 or 2 to insert right : 2

Enter 1 to insert left of = 7 or 2 to insert right : 1

The number of nodes of the tree is : 5
Opcount is : 11linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$ gcc pgm
2.c -o pgm2
linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$ 
```



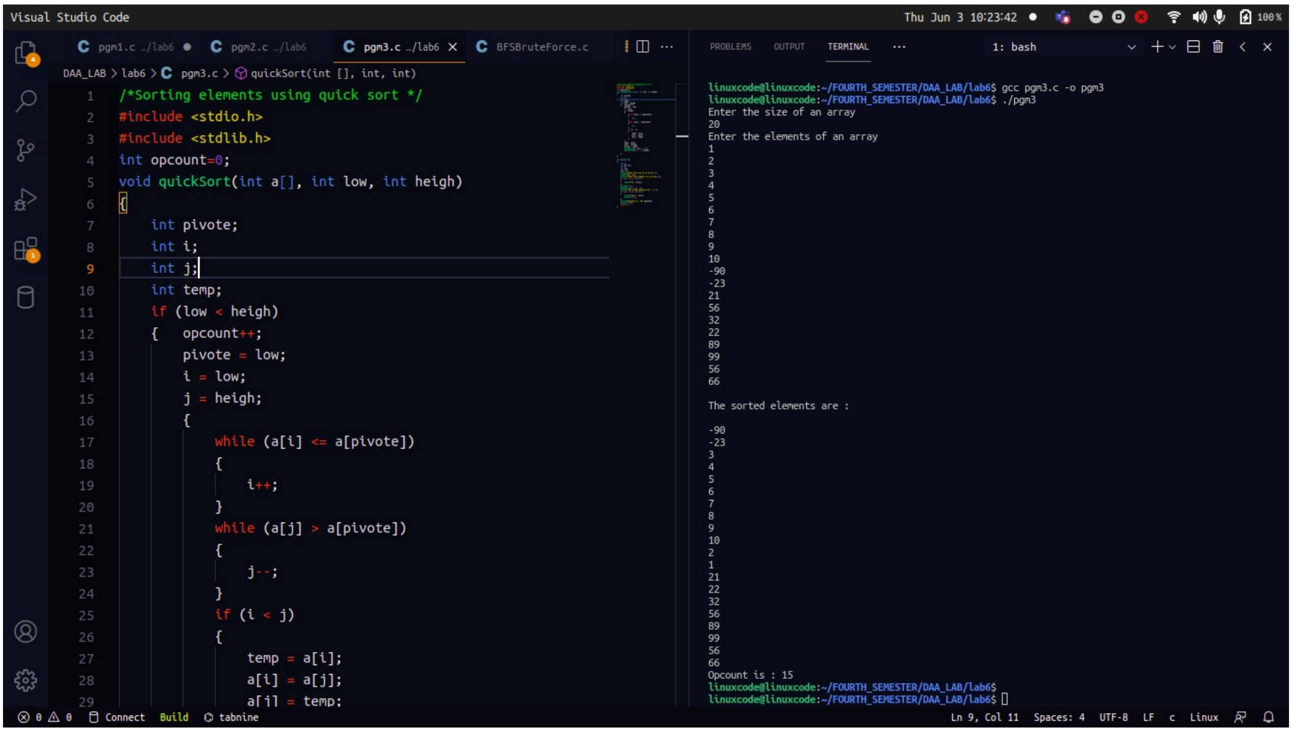
2) Sort given set of integers using Quick sort and analyze its efficiency. Obtain the experimental result of order of growth and plot the result.

```c
#include <stdio.h>
#include <stdlib.h>
int opcount=0;
void quickSort(int a[], int low, int heigh)
{
```

```
int pivote;
int i;
int j;
int temp;
if (low < heigh)
{ opcount++;
pivote = low;
i = low;
j = heigh;
{
while (a[i] <= a[pivote])
{
i++;
}
while (a[j] > a[pivote])
{
j--;
}
if (i < j)
{
temp = a[i];
a[i] = a[j];
a[j] = temp;
}
}
temp = a[j];
a[j] = a[low];
a[low] = temp;
quickSort(a, low, j - 1);
quickSort(a, j + 1, heigh);

}
}

int main(void)
{
int i;
int a[100];
int n;
int low;
low = i;
printf("Enter the size of an array\n");
```

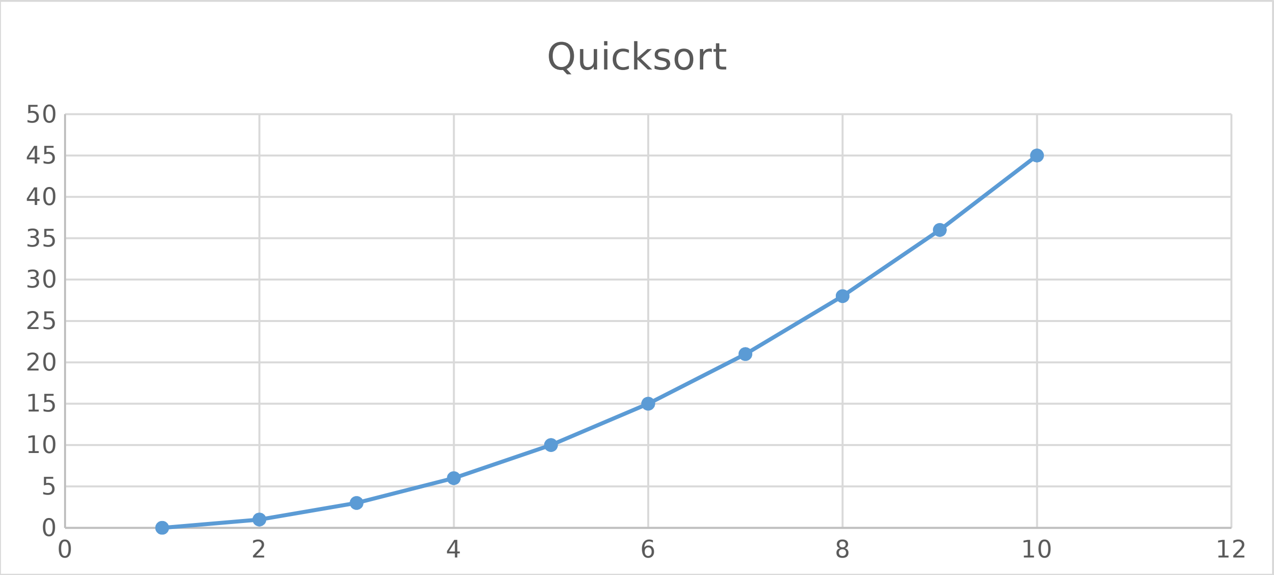
```c
scanf("%d", &n);
printf("Enter the elements of an array\n");
for (i = 0; i < n; i++)
{
scanf("%d", &a[i]);
}
printf("\n");
quickSort(a, 0, n - 1);
printf("The sorted elements are :\n\n");
for (i = 0; i < n; i++)
{
printf("%d\t", a[i]);
printf("\n");
}
printf("Opcount is : %d",opcount);
printf("\n");
return 0;
}
```

OUTPUT :



```c
/*Sorting elements using quick sort */
#include <stdio.h>
#include <stdlib.h>
int opcount=0;
void quickSort(int a[], int low, int heigh)
{
    int pivote;
    int i;
    int j;
    int temp;
    if (low < heigh)
    {   opcount++;
        pivote = low;
        i = low;
        j = heigh;
        {
            while (a[i] <= a[pivote])
            {
                i++;
            }
            while (a[j] > a[pivote])
            {
                j--;
            }
            if (i < j)
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
```

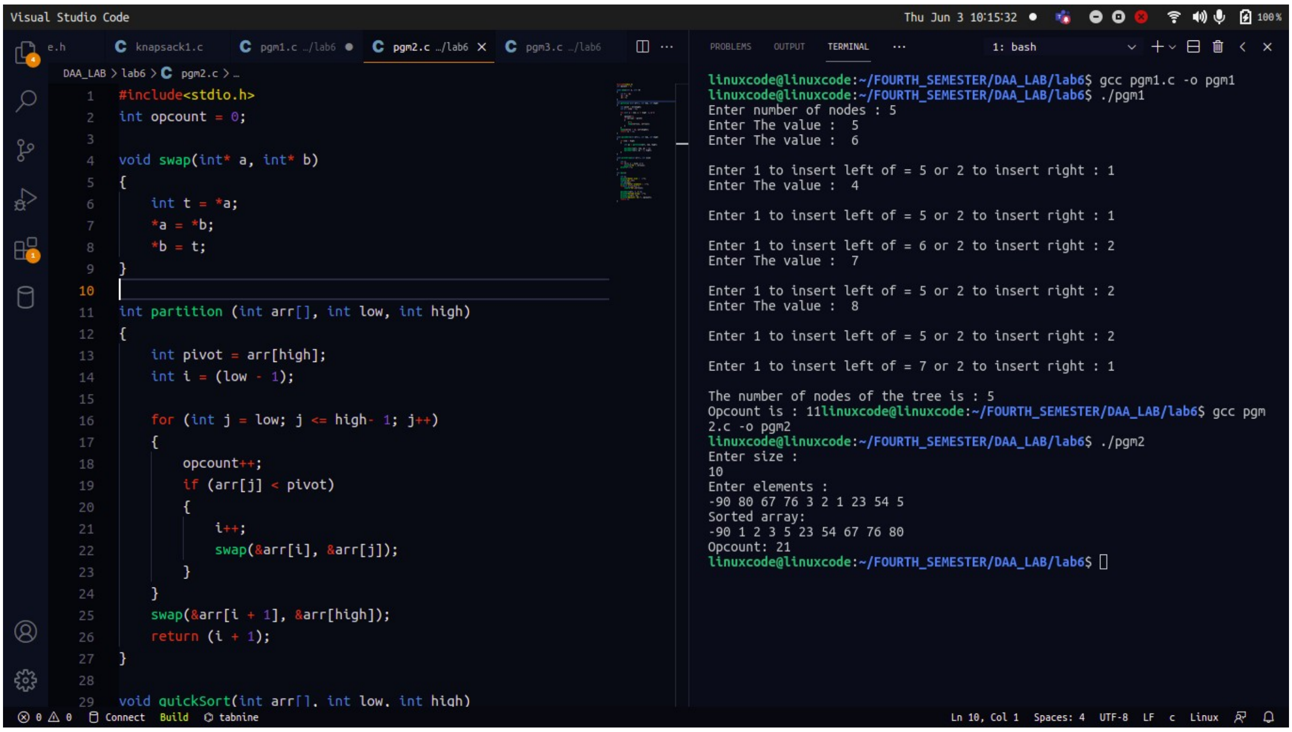
```
linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$ gcc pgm3.c -o pgm3
linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$ ./pgm3
Enter the size of an array
20
Enter the elements of an array
1
2
3
4
5
6
7
8
9
10
-90
-23
21
56
32
22
89
99
56
66

The sorted elements are :

-90
-23
3
4
5
6
7
8
9
10
2
1
21
22
32
56
89
99
56
66
Opcount is : 15
linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$
linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$
```



3) Sort given set of integers using Merge sort and analyze its efficiency. Obtain the experimental result of order of growth and plot the result.

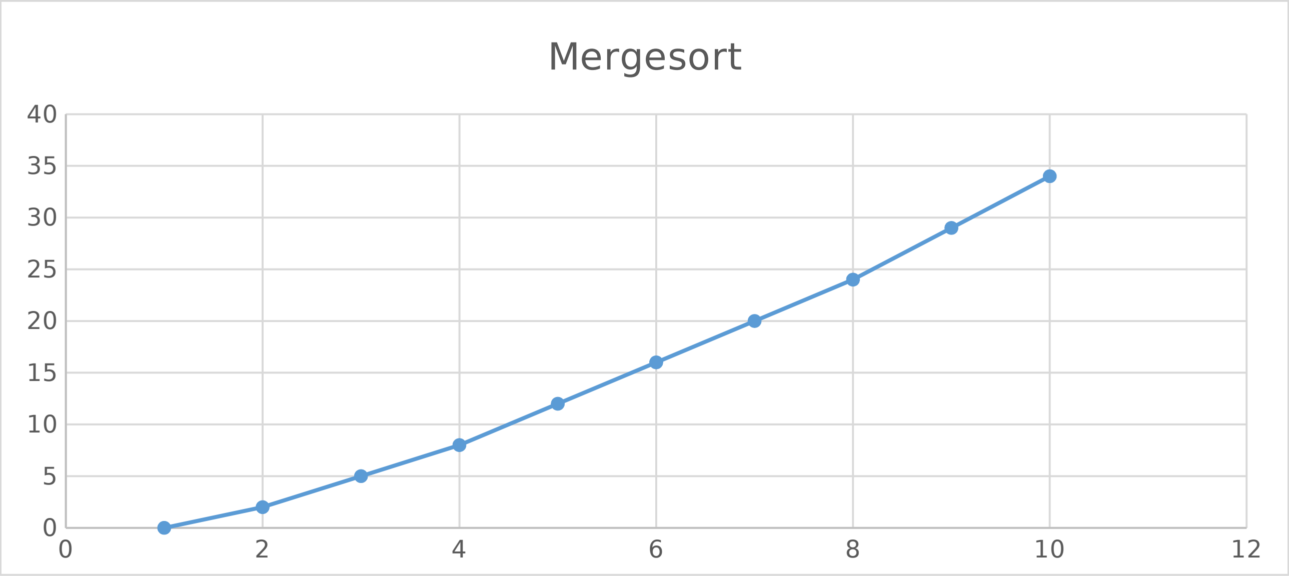
```
#include<stdio.h>
int opcount = 0;

void swap(int* a, int* b)
{
int t = *a;
*a = *b;
*b = t;
}

int partition (int arr[], int low, int high)
{
int pivot = arr[high];
int i = (low - 1);

for (int j = low; j <= high- 1; j++)
{
opcount++;
if (arr[j] < pivot)
{
i++;
swap(&arr[i], &arr[j]);
}
}
swap(&arr[i + 1], &arr[high]);
return (i + 1);
}

void mergeSort(int arr[], int low, int high)
{
if (low < high)
{
int pi = partition(arr, low, high);

mergeSort(arr, low, pi - 1);
mergeSort(arr, pi + 1, high);
}
}

void printArray(int arr[], int size)
{
int i;
for (i=0; i < size; i++)
printf("%d ", arr[i]);
printf("\n");
}

int main()
{
int n;
printf("Enter size : \n");
scanf("%d",&n);
int arr[n];
printf("Enter elements : \n");
for(int i=0;i<n;i++)
scanf("%d",&arr[i]);
mergeSort(arr, 0, n-1);
printf("Sorted array: \n");
printArray(arr, n);
printf("Opcount: %d\n", opcount);
return 0;
}
```

OUTPUT :



```c
#include<stdio.h>
int opcount = 0;

void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

int partition (int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high- 1; j++)
    {
        opcount++;
        if (arr[j] < pivot)
        {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void quickSort(int arr[], int low, int high)
```

```
linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$ gcc pgm1.c -o pgm1
linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$ ./pgm1
Enter number of nodes : 5
Enter The value :  5
Enter The value :  6

Enter 1 to insert left of = 5 or 2 to insert right : 1
Enter The value :  4

Enter 1 to insert left of = 5 or 2 to insert right : 1

Enter 1 to insert left of = 6 or 2 to insert right : 2
Enter The value :  7

Enter 1 to insert left of = 5 or 2 to insert right : 2
Enter The value :  8

Enter 1 to insert left of = 5 or 2 to insert right : 2

Enter 1 to insert left of = 7 or 2 to insert right : 1

The number of nodes of the tree is : 5
Opcount is : 11linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$ gcc pgm
2.c -o pgm2
linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$ ./pgm2
Enter size :
10
Enter elements :
-90 80 67 76 3 2 1 23 54 5
Sorted array:
-90 1 2 3 5 23 54 67 76 80
Opcount: 21
linuxcode@linuxcode:~/FOURTH_SEMESTER/DAA_LAB/lab6$ 
```


Mergesort