

LAB EXERCISE :

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;

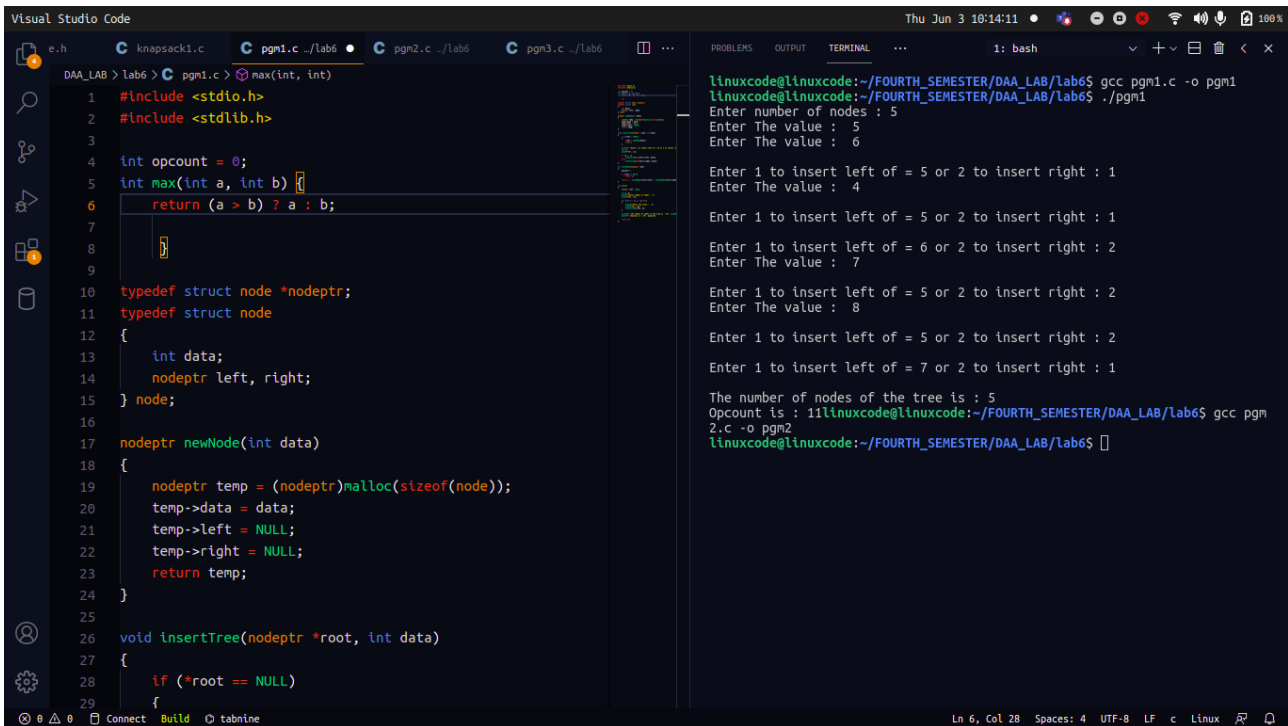
int n, m;
printf("Enter number of nodes : ");
scanf("%d", &n);

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 :



The screenshot shows a Visual Studio Code editor with a C program for inserting nodes into a binary tree. The code includes `<stdio.h>` and `<stdlib.h>`, defines a `node` structure with `data`, `left`, and `right` pointers, and implements a `newNode` function and an `insertTree` function. The terminal output shows the program being compiled and run, with user input for the number of nodes (5) and values (5, 6, 4, 7, 8) being inserted into the tree. The final output indicates that the number of nodes is 5 and the operation count is 11.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int opcount = 0;
5 int max(int a, int b) {
6     return (a > b) ? a : b;
7 }
8
9
10 typedef struct node *nodeptr;
11 typedef struct node
12 {
13     int data;
14     nodeptr left, right;
15 } node;
16
17 nodeptr newNode(int data)
18 {
19     nodeptr temp = (nodeptr)malloc(sizeof(node));
20     temp->data = data;
21     temp->left = NULL;
22     temp->right = NULL;
23     return temp;
24 }
25
26 void insertTree(nodeptr *root, int data)
27 {
28     if (*root == NULL)
29     {
```

```
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

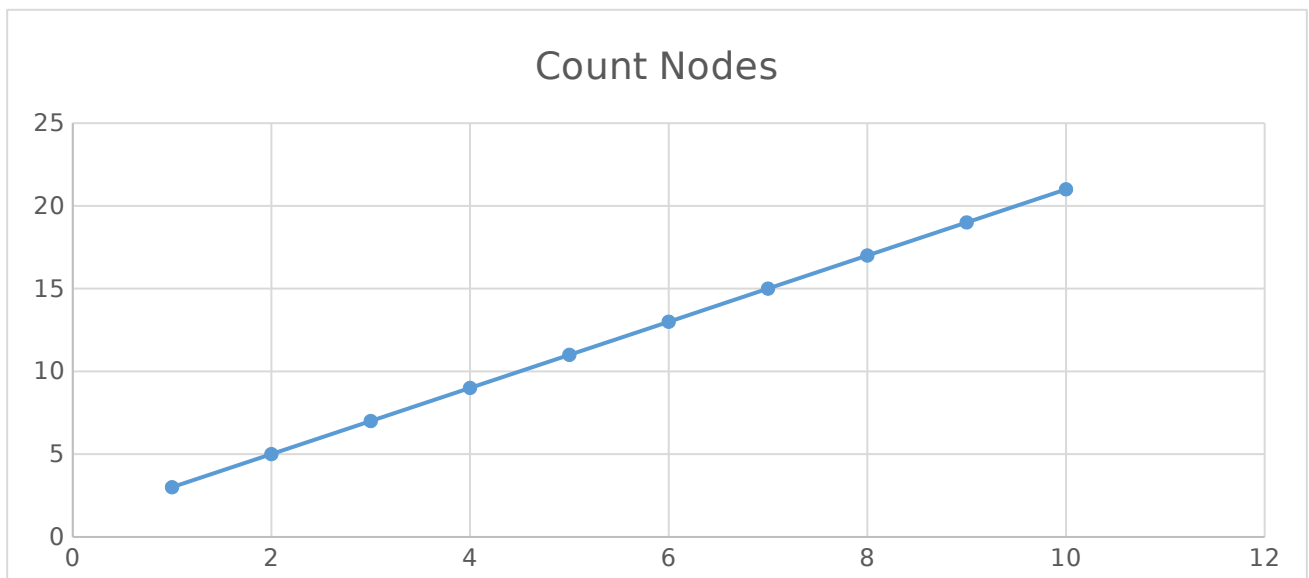
Enter 1 to insert left of = 5 or 2 to insert right : 1
Enter The value : 4

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 = 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.

```
#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");

```

```

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 :

The screenshot shows the Visual Studio Code interface with the following content:

Source Code (pgm3.c):

```

1  /*Sorting elements using quick sort */
2  #include <stdio.h>
3  #include <stdlib.h>
4  int opcount=0;
5  void quickSort(int a[], int low, int heigh)
6  {
7      int pivote;
8      int i;
9      int j;
10     int temp;
11     if (low < heigh)
12     {
13         opcount++;
14         pivote = low;
15         i = low;
16         j = heigh;
17         while (a[i] <= a[pivote])
18         {
19             i++;
20         }
21         while (a[j] > a[pivote])
22         {
23             j--;
24         }
25         if (i < j)
26         {
27             temp = a[i];
28             a[i] = a[j];
29             a[j] = temp;

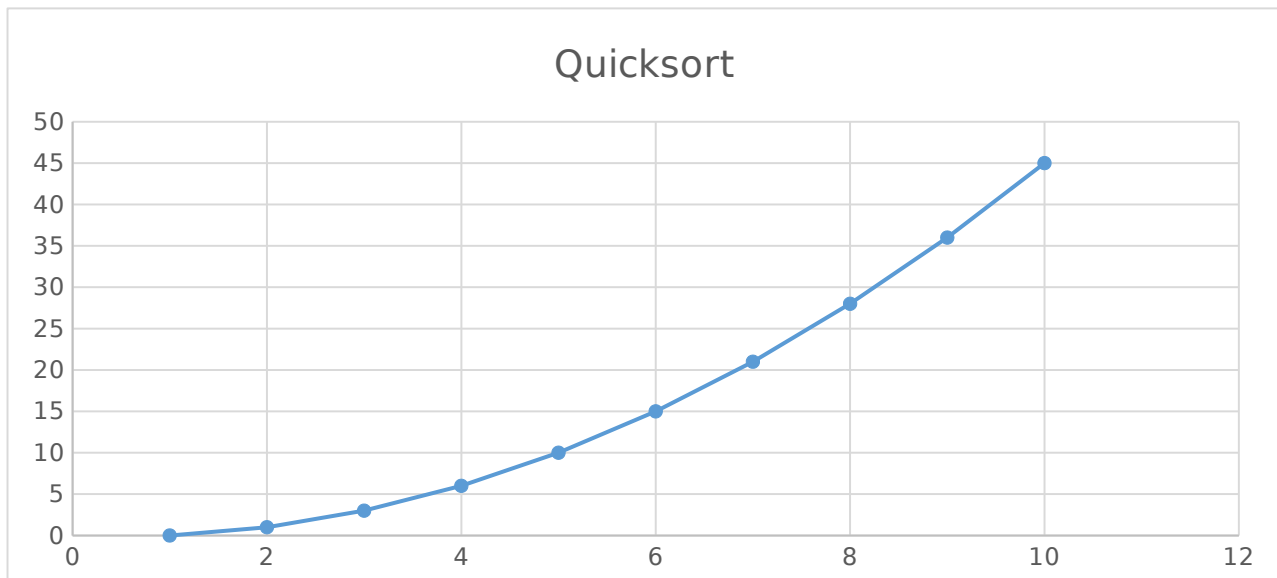
```

Terminal Output:

```

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
15
Enter the elements of an array
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
The sorted elements are :
-99
-23
3
4
5
6
7
8
9
10
11
12
13
14
15
Opcount is : 15
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 :

```
Visual Studio Code
DAA_LAB > lab6 > pgm2.c > ...
1 #include<stdio.h>
2 int opcount = 0;
3
4 void swap(int* a, int* b)
5 {
6     int t = *a;
7     *a = *b;
8     *b = t;
9 }
10
11 int partition (int arr[], int low, int high)
12 {
13     int pivot = arr[high];
14     int i = (low - 1);
15
16     for (int j = low; j <= high- 1; j++)
17     {
18         opcount++;
19         if (arr[j] < pivot)
20         {
21             i++;
22             swap(&arr[i], &arr[j]);
23         }
24     }
25     swap(&arr[i + 1], &arr[high]);
26     return (i + 1);
27 }
28
29 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

