

DAY 4

C PROGRAMMING – DATA STRUCTURE – 4

1.C programming to find binary transversal tree:

```
#include <stdio.h>

#include <stdlib.h>

struct node {
    int data;
    struct node *leftChild;
    struct node *rightChild;
};

struct node *root = NULL;

void insert(int data) {
    struct node *tempNode = (struct node*) malloc(sizeof(struct node));
    struct node *current;
    struct node *parent;
    tempNode->data = data;
    tempNode->leftChild = NULL;
    tempNode->rightChild = NULL;
    if(root == NULL) {
        root = tempNode;
    } else {
        current = root;
        parent = NULL;
        while(1) {
            parent = current;
            if(data < parent->data) {
                current = current->leftChild;
                if(current == NULL) {
                    parent->leftChild = tempNode;
                }
            }
        }
    }
}
```

```

        return;

    }

}

else {

    current = current->rightChild;

    if(current == NULL) {

        parent->rightChild = tempNode;

        return;

    }

}

}

}

}

```

```

        return NULL;
    }
}

return current;
}

void pre_order_traversal(struct node* root) {
    if(root != NULL) {
        printf("%d ",root->data);
        pre_order_traversal(root->leftChild);
        pre_order_traversal(root->rightChild);
    }
}

void inorder_traversal(struct node* root) {
    if(root != NULL) {
        inorder_traversal(root->leftChild);
        printf("%d ",root->data);
        inorder_traversal(root->rightChild);
    }
}

void post_order_traversal(struct node* root) {
    if(root != NULL) {
        post_order_traversal(root->leftChild);
        post_order_traversal(root->rightChild);
        printf("%d ", root->data);
    }
}

int main() {
    int i;

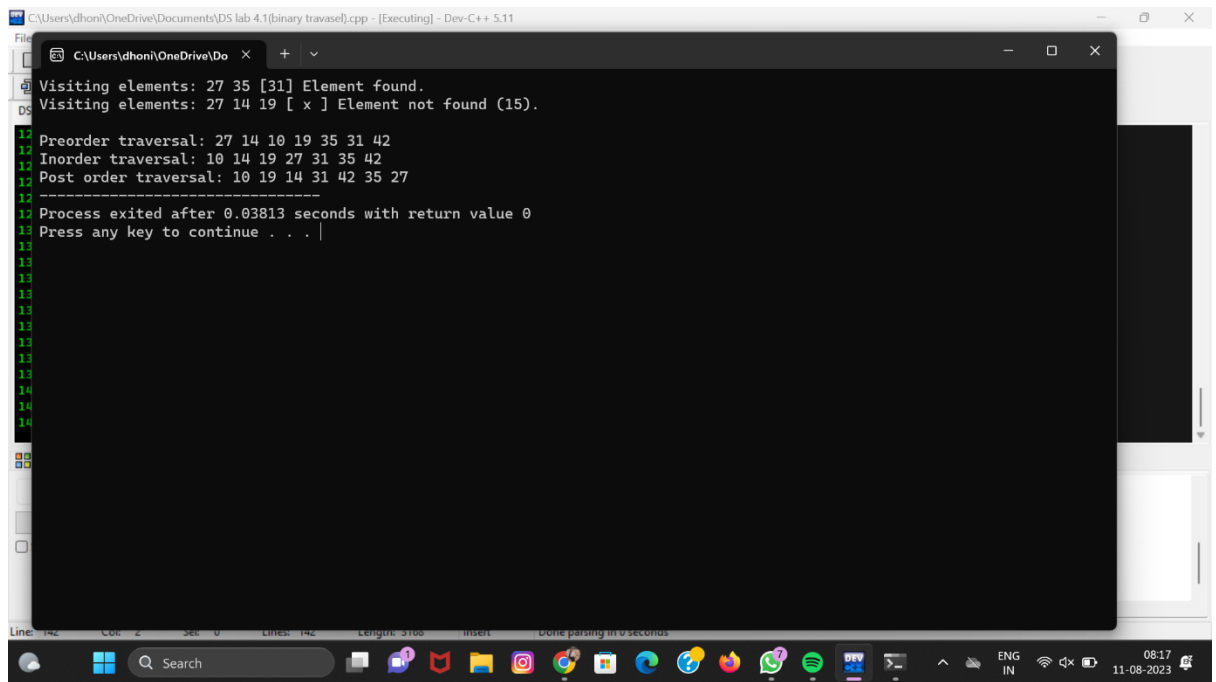
    int array[7] = { 27, 14, 35, 10, 19, 31, 42 };
    for(i = 0; i < 7; i++)

```

```

        insert(array[i]);
i = 31;
struct node * temp = search(i);
if(temp != NULL) {
    printf("[%d] Element found.", temp->data);
    printf("\n");
}else {
    printf("[ x ] Element not found (%d).\n", i);
}
i = 15;
temp = search(i);
if(temp != NULL) {
    printf("[%d] Element found.", temp->data);
    printf("\n");
}else {
    printf("[ x ] Element not found (%d).\n", i);
}
printf("\nPreorder traversal: ");
pre_order_traversal(root);
printf("\nInorder traversal: ");
inorder_traversal(root);
printf("\nPost order traversal: ");
post_order_traversal(root);
return 0;
}

```

A screenshot of a Windows desktop showing a Dev-C++ window. The window title is "C:\Users\dhoni\OneDrive\Documents\DS lab 4.1(binary travase).cpp - [Executing] - Dev-C++ 5.11". The console output shows the execution of an AVL tree program. It displays the insertion of elements 27, 35, and 31, followed by a search for 15 which fails. Then, it shows the preorder, inorder, and postorder traversals of the tree. The inorder traversal is 10 14 19 27 31 35 42. The program then exits after 0.03813 seconds.

```
C:\Users\dhoni\OneDrive\Documents\DS lab 4.1(binary travase).cpp - [Executing] - Dev-C++ 5.11
Visiting elements: 27 35 [31] Element found.
Visiting elements: 27 14 19 [ x ] Element not found (15).
Preorder traversal: 27 14 10 19 35 31 42
Inorder traversal: 10 14 19 27 31 35 42
Post order traversal: 10 19 14 31 42 35 27
-----
Process exited after 0.03813 seconds with return value 0
Press any key to continue . . .
```

2.C programming to implement AVL tree with all rotations:

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

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

```
struct Node {
```

```
    int key;
```

```
    struct Node *left;
```

```
    struct Node *right;
```

```
    int height;
```

```
};
```

```
int max(int a, int b);
```

```
int height(struct Node *N) {
```

```
    if (N == NULL)
```

```
        return 0;
```

```
    return N->height;
```

```
}
```

```
int max(int a, int b) {
```

```
    return (a > b) ? a : b;
```

```
}
```

```

struct Node *newNode(int key) {
    struct Node *node = (struct Node *)
        malloc(sizeof(struct Node));
    node->key = key;
    node->left = NULL;
    node->right = NULL;
    node->height = 1;
    return (node);
}

struct Node *rightRotate(struct Node *y) {
    struct Node *x = y->left;
    struct Node *T2 = x->right;
    x->right = y;
    y->left = T2;
    y->height = max(height(y->left), height(y->right)) + 1;
    x->height = max(height(x->left), height(x->right)) + 1;
    return x;
}

struct Node *leftRotate(struct Node *x) {
    struct Node *y = x->right;
    struct Node *T2 = y->left;
    y->left = x;
    x->right = T2;
    x->height = max(height(x->left), height(x->right)) + 1;
    y->height = max(height(y->left), height(y->right)) + 1;
    return y;
}

int getBalance(struct Node *N) {
    if (N == NULL)
        return 0;
    return height(N->left) - height(N->right);
}

```

```

}

struct Node *insertNode(struct Node *node, int key) {
    // Find the correct position to insertNode the node and insertNode it
    if (node == NULL)
        return (newNode(key));
    if (key < node->key)
        node->left = insertNode(node->left, key);
    else if (key > node->key)
        node->right = insertNode(node->right, key);
    else
        return node;
    node->height = 1 + max(height(node->left),
        height(node->right));
    int balance = getBalance(node);
    if (balance > 1 && key < node->left->key)
        return rightRotate(node);
    if (balance < -1 && key > node->right->key)
        return leftRotate(node);
    if (balance > 1 && key > node->left->key) {
        node->left = leftRotate(node->left);
        return rightRotate(node);
    }
    if (balance < -1 && key < node->right->key) {
        node->right = rightRotate(node->right);
        return leftRotate(node);
    }
    return node;
}

struct Node *minValueNode(struct Node *node) {
    struct Node *current = node;

```

```

while (current->left != NULL)
    current = current->left;
return current;
}

struct Node *deleteNode(struct Node *root, int key) {
    if (root == NULL)
        return root;
    if (key < root->key)
        root->left = deleteNode(root->left, key);
    else if (key > root->key)
        root->right = deleteNode(root->right, key);
    else {
        if ((root->left == NULL) || (root->right == NULL)) {
            struct Node *temp = root->left ? root->left : root->right;
            if (temp == NULL) {
                temp = root;
                root = NULL;
            } else
                *root = *temp;
            free(temp);
        } else {
            struct Node *temp = minValueNode(root->right);
            root->key = temp->key;
            root->right = deleteNode(root->right, temp->key);
        }
    }
    if (root == NULL)
        return root;
    root->height = 1 + max(height(root->left),
        height(root->right));
}

```



```

int balance = getBalance(root);
if (balance > 1 && getBalance(root->left) >= 0)
    return rightRotate(root);
if (balance > 1 && getBalance(root->left) < 0) {
    root->left = leftRotate(root->left);
    return rightRotate(root);
}
if (balance < -1 && getBalance(root->right) <= 0)
    return leftRotate(root);
if (balance < -1 && getBalance(root->right) > 0) {
    root->right = rightRotate(root->right);
    return leftRotate(root);
}
return root;
}

void printPreOrder(struct Node *root) {
    if (root != NULL) {
        printf("%d ", root->key);
        printPreOrder(root->left);
        printPreOrder(root->right);
    }
}

int main() {
    struct Node *root = NULL;
    root = insertNode(root, 2);
    root = insertNode(root, 1);
    root = insertNode(root, 7);
    root = insertNode(root, 4);
    root = insertNode(root, 5);
    root = insertNode(root, 3);
    root = insertNode(root, 8);
}

```

```

printPreOrder(root);

root = deleteNode(root, 3);

printf("\nAfter deletion: ");

printPreOrder(root);

return 0;

}

```

```

C:\Users\dhoni\OneDrive\Documents\DS lab 4.2\avl tree\avl tree.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.3 64-bit Release
DS lab 4.1(b
4 2 1 3 7 5 8
After deletion: 4 2 1 7 5 8
-----
Process exited after 0.07815 seconds with return value 0
Press any key to continue . . .
Line: 195

```

3.C programming to implement hashing using linear probing technique:

```

#include <stdio.h>

#include<stdlib.h>

#define TABLE_SIZE 10

int h[TABLE_SIZE]={NULL};

void insert()

{

    int key,index,i,flag=0,hkey;

    printf("\nenter a value to insert into hash table\n");

    scanf("%d",&key);

    hkey=key%TABLE_SIZE;

```

```

for(i=0;i<TABLE_SIZE;i++)
{
    index=(hkey+i)%TABLE_SIZE;
    if(h[index] == NULL)
    {
        h[index]=key;
        break;
    }
}
if(i == TABLE_SIZE)
    printf("\nelement cannot be inserted\n");
}

void search()
{
    int key,index,i,flag=0,hkey;
    printf("\nEnter search element\n");
    scanf("%d",&key);
    hkey=key%TABLE_SIZE;
    for(i=0;i<TABLE_SIZE; i++)
    {
        index=(hkey+i)%TABLE_SIZE;
        if(h[index]==key)
        {
            printf("value is found at index %d",index);
            break;
        }
    }
    if(i == TABLE_SIZE)
        printf("\n value is not found\n");
}

void display()

```

```

{

    int i;

    printf("\nelements in the hash table are \n");

    for(i=0;i< TABLE_SIZE; i++)

        printf("\nat index %d \t value = %d",i,h[i]);

}

main()
{
    int opt,i;

    while(1)

    {

        printf("\nPress 1. Insert\t 2. Display \t3. Search \t4.Exit \n");

        scanf("%d",&opt);

        switch(opt)

        {

            case 1:

                insert();

                break;

            case 2:

                display();

                break;

            case 3:

                search();

                break;

            case 4:exit(0);

        }

    }

}

```

```
C:\Users\dhoni\OneDrive\Do x + v
Press 1. Insert 2. Display 3. Search 4.Exit
1
enter a value to insert into hash table
23
Press 1. Insert 2. Display 3. Search 4.Exit
1
enter a value to insert into hash table
45
Press 1. Insert 2. Display 3. Search 4.Exit
1
enter a value to insert into hash table
60
Press 1. Insert 2. Display 3. Search 4.Exit
2
elements in the hash table are
at index 0 value = 60
at index 1 value = 0
at index 2 value = 0
at index 3 value = 23
at index 4 value = 0
at index 5 value = 45
at index 6 value = 0
at index 7 value = 0
at index 8 value = 0
at index 9 value = 0
Press 1. Insert 2. Display 3. Search 4.Exit
3
enter search element
23
value is found at index 3
```

4.C programming to implement :

1.bubble sorting :

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

```
int main(){
```

```
    int a[50], i,j,n,t;
```

```
    printf("enter the No: of elements in the list:");
```

```
    scanf("%d", &n);
```

```
    printf("enter the elements:");
```

```
    for(i=0; i<n; i++){
```

```
        scanf ("%d", &a[i]);
```

```
    }
```

```
    printf("Before bubble sorting the elements are:");
```

```
    for(i=0; i<n; i++)
```

```
        printf("%d \t", a[i]);
```

```
    for (i=0; i<n-1; i++){
```

```
        for (j=i+1; j<n; j++){
```

```
            if (a[i] > a[j]){
```

```
                t = a[i];
```

```

        a[i] = a[j];

        a[j] = t;

    }

}

printf("\nafter bubble sorting the elements are:");

for (i=0; i<n; i++)

    printf("%d\t", a[i]);

return 0;

}

```

The screenshot shows the Dev-C++ IDE with the following code in the editor:

```

8   scanf("%d", &n);
9   int a[n];
10  printf("\nenter the No: of elements in the list:");
11  for(i=0; i<n; i++)
12  {
13      printf("enter the elements:");
14      for(j=0; j<n; j++)
15      {
16          scanf("%d", &a[j]);
17      }
18      printf("\nBefore bubble sorting the elements are:");
19      for(i=0; i<n; i++)
20      {
21          printf("%d\t", a[i]);
22      }
23      printf("\nafter bubble sorting the elements are:");
24      for(i=0; i<n; i++)
25      {
26          printf("%d\t", a[i]);
27      }
28  }
29  return 0;
30  }

```

The output window shows the following execution results:

```

C:\Users\dhoni\OneDrive\Documents\DS lab 4.5.1(bubble sort).cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
DS lab 4.1(binary travasel).cpp DS lab 4.2(avl tree).cpp DS lab 4.3 (hashing).cpp DS lab 4.5.1(bubble sort).cpp
C:\Users\dhoni\OneDrive\Documents\DS lab 4.5.1(bubble sort).cpp
8   scanf("%d", &n);
9   int a[n];
10  printf("\nenter the No: of elements in the list:");
11  for(i=0; i<n; i++)
12  {
13      printf("enter the elements:");
14      for(j=0; j<n; j++)
15      {
16          scanf("%d", &a[j]);
17      }
18      printf("\nBefore bubble sorting the elements are:");
19      for(i=0; i<n; i++)
20      {
21          printf("%d\t", a[i]);
22      }
23      printf("\nafter bubble sorting the elements are:");
24      for(i=0; i<n; i++)
25      {
26          printf("%d\t", a[i]);
27      }
28  }
29  return 0;
30  }
Compiler
Abort Compilation
Shorten compiler path
Line: 25 Col: 1
30°
ENG IN
09:15
11-08-2023

```

2.selection sorting:

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

```
int main()
```

```
{
```

```
int array[100], n, c, d, position, t;
```

```
printf("Enter number of elements\n");
```

```
scanf("%d", &n);
```

```
printf("Enter %d integers\n", n);
```

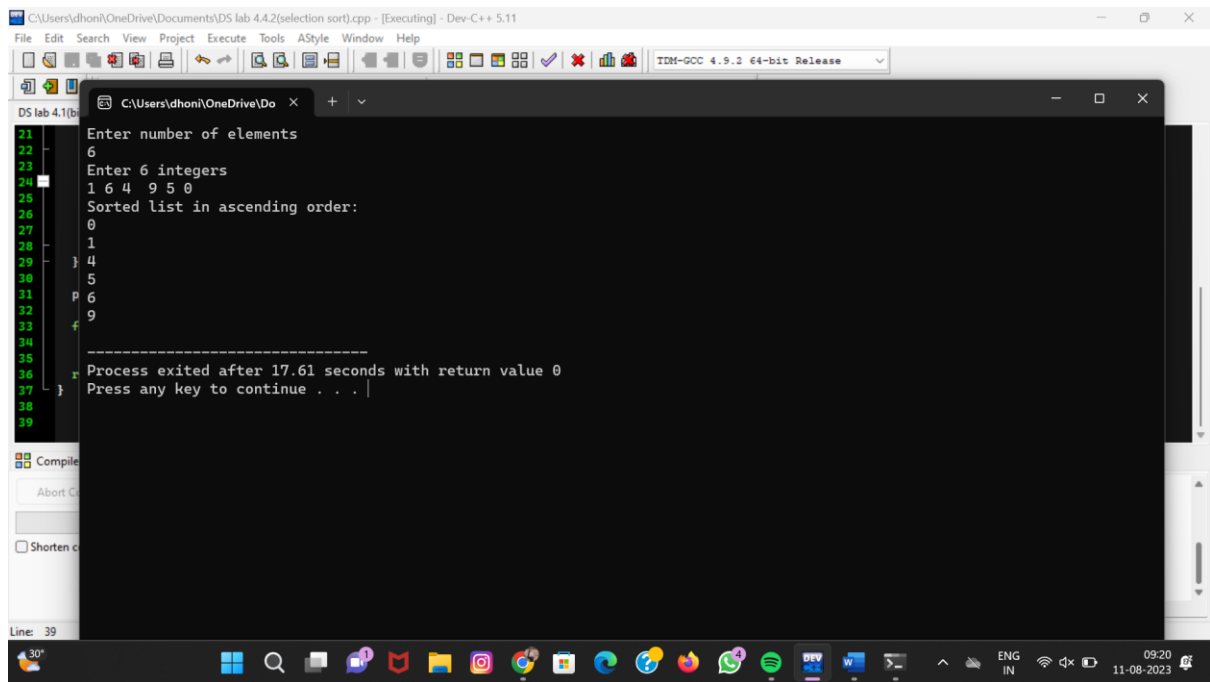
```
for (c = 0; c < n; c++)
```

```

scanf("%d", &array[c]);
for (c = 0; c < (n - 1); c++) // finding minimum element (n-1) times
{
    position = c;
    for (d = c + 1; d < n; d++)
    {
        if (array[position] > array[d])
            position = d;
    }
    if (position != c)
    {
        t = array[c];
        array[c] = array[position];
        array[position] = t;
    }
}

printf("Sorted list in ascending order:\n");
for (c = 0; c < n; c++)
    printf("%d\n", array[c]);
return 0;
}

```



```
C:\Users\dhoni\OneDrive\Documents\DS lab 4.4.2(selection sort).cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release

DS lab 4.1(b
21 Enter number of elements
22 6
23 Enter 6 integers
24 1 6 4 9 5 0
25 Sorted list in ascending order:
26 0
27 1
28 4
29 5
30 6
31 9
32
33 -----
34 Process exited after 17.61 seconds with return value 0
35 Press any key to continue . . . |
36
37
38
39
Line: 39
```

3.insertion sort :

```
#include <math.h>
```

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

```
void insertionSort(int arr[], int n)
```

```
{
```

```
    int i, key, j;
```

```
    for (i = 1; i < n; i++)
```

```
    {
```

```
        key = arr[i];
```

```
        j = i - 1;
```

```
        while (j >= 0 && arr[j] > key)
```

```
        {
```

```
            arr[j + 1] = arr[j];
```

```
            j = j - 1;
```

```
        }
```

```
        arr[j + 1] = key;
```

```
    }
```

```
}
```



```

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

int main()
{
    int arr[] = {12, 11, 13, 5, 6};
    int n = sizeof(arr) / sizeof(arr[0]);
    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}

```

The screenshot shows a Dev-C++ IDE window titled "C:\Users\dhoni\OneDrive\Documents\DS lab 4.4.3.cpp - [Executing] - Dev-C++ 5.11". The code editor displays the following C++ code:

```

11 {
12     int i;
13     for (i = 0; i < n; i++)
14         printf("%d ", arr[i]);
15     printf("\n");
16 }
17
18 // Driver
19 int main()
20 {
21     int arr[] = {12, 11, 13, 5, 6};
22     int n = sizeof(arr) / sizeof(arr[0]);
23     insertionSort(arr, n);
24     printArray(arr, n);
25     return 0;
26 }

```

The output window shows the following output:

```

5 6 11 12 13
Process exited after 0.03835 seconds with return value 0
Press any key to continue . . .

```

The taskbar at the bottom shows the date and time as 11-08-2023, 09:22.

4.quick sort :

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

```
void quicksort(int number[25],int first,int last){
```

```
    int i, j, pivot, temp;
```

```

if(first<last){
    pivot=first;
    i=first;
    j=last;
    while(i<j){
        while(number[i]<=number[pivot]&& i<last)
            i++;
        while(number[j]>number[pivot])
            j--;
        if(i<j){
            temp=number[i];
            number[i]=number[j];
            number[j]=temp;
        }
    }
    temp=number[pivot];
    number[pivot]=number[j];
    number[j]=temp;
    quicksort(number,first,j-1);
    quicksort(number,j+1,last);
}
}

int main(){
    int i, count, number[25];
    printf("How many elements are u going to enter?: ");
    scanf("%d",&count);
    printf("Enter %d elements: ", count);
    for(i=0;i<count;i++)
        scanf("%d",&number[i]);
    quicksort(number,0,count-1);
    printf("Order of Sorted elements: ");

```

```

for(i=0;i<count;i++)

printf("%d",number[i]);

return 0;

}

```

```

1 #include<stdio.h>
2 void quicksort(int number[],int first,int last){
3     int i, j;
4     if(first<last)
5     {
6         pivot=
7         i=first;
8         j=last;
9         while(i<j)
10        {
11            i++;
12            while(number[i]<=number[j])
13            {
14                i++;
15            }
16            while(number[j]>=number[i])
17            {
18                j--;
19            }
20            temp=number[i];
21            number[i]=number[j];
22            number[j]=temp;
23        }
24        quicksort(number,i,j);
25        quicksort(number,j+1,i);
26    }
27 }
28 int main()
29 {
30     int number[100],i,j;
31     int first,last;
32     printf("How many elements are u going to enter?: ");
33     scanf("%d",&first);
34     printf("Enter %d elements: ",first);
35     for(i=0;i<first;i++)
36     {
37         scanf("%d",&number[i]);
38     }
39     quicksort(number,0,first-1);
40     printf("Order of Sorted elements: ");
41     for(i=0;i<first;i++)
42     {
43         printf("%d ",number[i]);
44     }
45     printf("\n");
46     return 0;
47 }

```

5. Merge sort:

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

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

```
void merge(int arr[], int l, int m, int r)
```

```
{
```

```
    int i, j, k;
```

```
    int n1 = m - l + 1;
```

```
    int n2 = r - m;
```

```
    int L[n1], R[n2];
```

```
    for (i = 0; i < n1; i++)
```

```
        L[i] = arr[l + i];
```

```
    for (j = 0; j < n2; j++)
```

```
        R[j] = arr[m + 1 + j];
```

```
    i = 0;
```

```

j = 0;
k = l;
while (i < n1 && j < n2) {
    if (L[i] <= R[j]) {
        arr[k] = L[i];
        i++;
    }
    else {
        arr[k] = R[j];
        j++;
    }
    k++;
}
while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
}
while (j < n2) {
    arr[k] = R[j];
    j++;
    k++;
}
}

void mergeSort(int arr[], int l, int r)
{
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
    }
}

```

```
        merge(arr, l, m, r);
    }
}

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

int main()
{
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    int arr_size = sizeof(arr) / sizeof(arr[0]);
    printf("Given array is \n");
    printArray(arr, arr_size);
    mergeSort(arr, 0, arr_size - 1);
    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}
```

```
C:\Users\dhoni\OneDrive\Documents\DS lab 4.5.5(merg sort).cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help

C:\Users\dhoni\OneDrive\Do x + v - _ □ X

DS lab- Given array is
1 12 11 13 5 6 7
2
3 Sorted array is
4 5 6 7 11 12 13
5
6 -----
7 Process exited after 0.03987 seconds with return value 0
8 Press any key to continue . . .
9
10
11
12
13
14
15
16
17
18
19
20
Line: 13
```

30°

20:22
11-08-2023