WEEK-5

DATA AND ANALYSIS ALGORITHM

NAME:KANTAMRAJU SREEJA

ROLLNO:CH.SC.U4CSE24255

1)QUICK SORT:

CODE:

```c
#include <stdio.h>

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

    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    return (i + 1);
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

```c
int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    printf("Given array: \n");
    printArray(arr, n);

    quickSort(arr, 0, n - 1);
    printf("\nSorted array: \n");
    printArray(arr, n);
    return 0;
}
```

OUTPUT:

```
kantamraju@kantamraju-VirtualBox:~$ gcc quicksort.c -o quicksort
kantamraju@kantamraju-VirtualBox:~$ ./quicksort
Enter the number of elements: 10
Enter the elements of the array:
31 58 23 10 30 49 50 24 22 51
Given array:
31 58 23 10 30 49 50 24 22 51

Sorted array:
10 22 23 24 30 31 49 50 51 58
```

2)MERGE SORT:

```c
#include <stdio.h>
void merge(int arr[], int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[n1], R[n2];

    for (int i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (int i = 0; i < n2; i++)
        R[i] = arr[m + 1 + i];

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

int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];

    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Given array: \n");
    printArray(arr, n);


    mergeSort(arr, 0, n - 1);

    printf("\nSorted array: \n");
    printArray(arr, n);

    return 0;
}
```

OUTPUT:

```
kantamraju@kantamraju-VirtualBox:~$ gcc mergesort.c -o mergesort
kantamraju@kantamraju-VirtualBox:~$ ./mergesort
Enter the number of elements: 10
Enter the elements of the array:
31 58 23 10 30 49 50 24 22 51
Given array:
31 58 23 10 30 49 50 24 22 51

Sorted array:
10 22 23 24 30 31 49 50 51 58
```

3) BST (BINARY SEARCH TREE):

CODE:

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}
struct Node* insert(struct Node* root, int value) {
    if (root == NULL)
        return createNode(value);

    if (value < root->data)
        root->left = insert(root->left, value);
    else if (value > root->data)
        root->right = insert(root->right, value);

    return root;
}
void inorder(struct Node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}
```

```c
int search(struct Node* root, int key) {
    if (root == NULL)
        return 0;
    if (root->data == key)
        return 1;
    if (key < root->data)
        return search(root->left, key);
    return search(root->right, key);
}
int main() {
    struct Node* root = NULL;
    root = insert(root, 50);
    insert(root, 30);
    insert(root, 70);
    insert(root, 20);
    insert(root, 40);
    insert(root, 60);
    insert(root, 80);
    printf("Inorder Traversal: ");
    inorder(root);
    printf("\nPreorder Traversal: ");
    preorder(root);
    printf("\nPostorder Traversal: ");
    postorder(root);
    int key = 40;
    if (search(root, key))
        printf("\n%d found in BST", key);
    else
        printf("\n%d not found in BST", key);
    return 0;
}
```

Output:

```
kantamraju@kantamraju-VirtualBox:~$ gcc bst.c -o bst
kantamraju@kantamraju-VirtualBox:~$ ./bst
Enter the number of elements: 10
Enter the elements of the array:
31 58 23 10 30 49 50 24 22 51
Given array:
31 58 23 10 30 49 50 24 22 51

Sorted array:
10 22 23 24 30 31 49 50 51 58
```