Nithin S
221IT085

IT206 Lab Assignment

Q1) Implement Quick Sort

CODE

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

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

int partition(int A[],int l,int h)
{
    int i=l;
    int j=h;
    int pivot=A[l];
    do
    {
        do{i++;}while(A[i]<=pivot);
        do{j--;}while(A[j]>pivot);
        if(i<j) swap(&A[i],&A[j]);
    }while(i<j);
    swap(&A[l],&A[j]);
    return j;
}
```

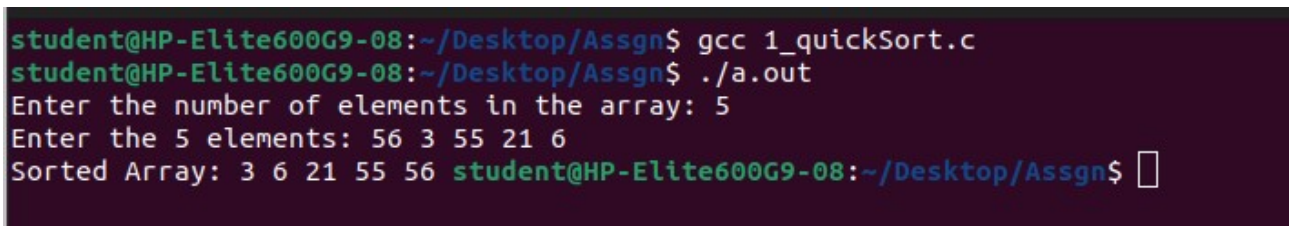
```
void quickSort(int A[],int l,int h)
{
    int j;
    if(l<h)
    {
        j=partition(A,l,h);
        quickSort(A,l,j);
        quickSort(A,j+1,h);
    }
}


int main()
{
    int n = 0;
    printf("Enter the number of elements in the
array: ");
    scanf("%d", &n);
    int *arr = (int *)malloc(sizeof(int) * n);
    printf("Enter the %d elements: ", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    quickSort(arr, 0, n - 1);

    printf("Sorted Array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    free(arr);
    return 0;
}
```

OUTPUT



Q2) Implement Preorder, PostOrder and Inorder in Binary Tree

CODE

Queue.h

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

struct Node
{
    struct Node *lchild;
    int data;
    struct Node *rchild;
};

struct Queue
{
    int size;
    int front;
    int rear;
    struct Node **Q;
};

void create(struct Queue *q, int size)
{
    q->size = size;
    q->front = q->rear = 0;
    q->Q = (struct Node **)malloc(q->size *
sizeof(struct Node *));
}

void enqueue(struct Queue *q, struct Node *x)
{
    if ((q->rear + 1) % q->size == q->front)
        printf("Queue is Full");
    else
    {
```

```
        q->rear = (q->rear + 1) % q->size;
        q->Q[q->rear] = x;
    }
}
struct Node *dequeue(struct Queue *q)
{
    struct Node *x = NULL;

    if (q->front == q->rear)
        printf("Queue is Empty\n");
    else
    {
        q->front = (q->front + 1) % q->size;
        x = q->Q[q->front];
    }
    return x;
}
int isEmpty(struct Queue q)
{
    return q.front == q.rear;
}
```

BT.c

```
#include <stdio.h>
#include <stdlib.h>
#include "Queue.h"
struct Node *root = NULL;

void Treecreate()
{
    struct Node *p, *t;
    int x;
    struct Queue q;
    create(&q, 100);

    printf("Eneter root value: ");
    scanf("%d", &x);
    root = (struct Node *)malloc(sizeof(struct
Node));
```

```
    root->data = x;
    root->lchild = root->rchild = NULL;
    enqueue(&q, root);
    while (!isEmpty(q))
    {
        p = dequeue(&q);
        printf("Enter left child of %d: ", p->data);
        scanf("%d", &x);
        if (x != -1)
        {
            t = (struct Node *)malloc(sizeof(struct
                                              Node));
            t->data = x;
            t->lchild = t->rchild = NULL;
            p->lchild = t;
            enqueue(&q, t);
        }
        printf("Enter right child of %d: ", p->data);
        scanf("%d", &x);
        if (x != -1)
        {
            t = (struct Node *)malloc(sizeof(struct
                                              Node));
            t->data = x;
            t->lchild = t->rchild = NULL;
            p->rchild = t;
            enqueue(&q, t);
        }
    }
}
void Preorder(struct Node *p)
{
    if (p)
    {
        printf("%d ", p->data);
        Preorder(p->lchild);
        Preorder(p->rchild);
    }
}
void Inorder(struct Node *p)
{
```

```
    if (p)
    {
        Inorder(p->lchild);
        printf("%d ", p->data);
        Inorder(p->rchild);
    }
}
void Postorder(struct Node *p)
{
    if (p)
    {
        Postorder(p->lchild);
        Postorder(p->rchild);
        printf("%d ", p->data);
    }
}
int main()
{
    Treecreate();
    printf("Pre Order: ");
    Preorder(root);
    printf("\nPost Order ");
    Postorder(root);
    printf("\nIn Order ");
    Inorder(root);

    return 0;
}
```

OUTPUT

```
student@HP-Elite600G9-08:~/Desktop/Assgn$ gcc 2_BInaryTree.c
student@HP-Elite600G9-08:~/Desktop/Assgn$ ./a.out
Eneter root value: 45
Enter left child of 45: 32
Enter right child of 45: 67
Enter left child of 32: 87
Enter right child of 32: 9
Enter left child of 67: 32
Enter right child of 67: 45
Enter left child of 87: -1
Enter right child of 87: -1
Enter left child of 9: -1
Enter right child of 9: -1
Enter left child of 32: -1
Enter right child of 32: -1
Enter left child of 45: -1
Enter right child of 45: -1
Pre Order: 45 32 87 9 67 32 45
Post Order 87 9 32 32 45 67 45
In Order 87 32 9 45 32 67 45 student@HP-Elite600G9-08:~/Desktop/Assgn$
```