1. Write a C program that uses functions to perform the following:
a) Create a singly linked list of integers.
b) Delete a given integer from the above linked list.
c) Display the contents of the above list after deletion.

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

struct node{
    int data;
    struct node* link;
};

struct node* createList(struct node*);
struct node* deleteNode(struct node*);
void display(struct node*);

int main(){
    struct node *head = NULL;
    int choice = 1;
    int option;

    head = createList(head);
    printf("After creation list is:\n");
    display(head);
    while(choice){
        printf("\n1.Delete A Node\n2.Exit\n");
        printf("Enter you choice: ");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                head = deleteNode(head);
                printf("After deletion list is: \n");
                display(head);
                break;
            case 2:
                choice = 0;
                break;
        }
    }
    return 0;
}

struct node* createList(struct node* head){
    struct node *temp = NULL,*new = NULL;
    int choice = 1;
    while(choice){
        new = (struct node*)malloc(sizeof(struct node));
        printf("Enter the data: ");
        scanf("%d",&new->data);
        new->link = NULL;
        if(head == NULL){
            head = temp = new;
        }
        else{
            temp->link = new;
            temp = new;
        }
        printf("Do you want to add another element (0/1): ");
        scanf("%d",&choice);
    }
    return head;
}

struct node* deleteNode(struct node* head){
    int ele,pos;
    struct node *temp = head,*ptr = NULL;
    printf("\nEnter your element to be deleted: ");
    scanf("%d",&ele);
    while(temp->data != ele){
        ptr = temp;
        temp = temp->link;
    }
    if(temp == head){
        head = head->link;
    }
    else if(temp->link == NULL){
        ptr->link = NULL;
    }
    else{
        ptr->link = temp->link;
    }
    free(temp);
    temp = NULL;
    return head;
}

void display(struct node *head){
    struct node* temp = head;
    while(temp != NULL){
        printf("%d ",temp->data);
        temp = temp->link;
    }
    printf("\n");
}
```

OUTPUT :-

```
Enter the data: 21
Do you want to add another element (0/1): 1
Enter the data: 22
Do you want to add another element (0/1): 1
Enter the data: 23
Do you want to add another element (0/1): 1
Enter the data: 24
Do you want to add another element (0/1): 1
```

```
Enter the data: 25
Do you want to add another element (0/1): 1
Enter the data: 26
Do you want to add another element (0/1): 0
After creation list is:
21 22 23 24 25 26

1.Delete A Node
2.Exit
Enter you choice: 1
Enter your element to be deleted: 21
After deletion list is:
22 23 24 25 26

1.Delete A Node
2.Exit
Enter you choice: 1
Enter your element to be deleted: 26
After deletion list is:
22 23 24 25

1.Delete A Node
2.Exit
Enter you choice: 1
Enter your element to be deleted: 23
After deletion list is:
22 24 25

1.Delete A Node
2.Exit
Enter you choice: 2
```

2. Write a C program that uses functions to perform the following:
a) Create a doubly linked list of integers.
b) Delete a given integer from the above doubly linked list.
c) Display the contents of the above list after deletion.

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

struct node{
    int data;
    struct node* prev;
    struct node* next;
};

struct node* createList(struct node*);
struct node* deleteNode(struct node*);
void display(struct node*);

int main(){
    struct node *head = NULL;
    int choice = 1;
    int option;

    head = createList(head);
    printf("After creation list is:\n");
    display(head);
    while(choice){
        printf("\n1.Delete A Node\n2.Exit\n");
        printf("Enter you choice: ");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                head = deleteNode(head);
                printf("After deletion list is: \n");
                display(head);
                break;
            case 2:
                choice = 0;
                break;
        }
    }
    return 0;
}

struct node* createList(struct node* head){
    struct node *temp = NULL,*new = NULL;
    int choice = 1;
    while(choice){
        new = (struct node*)malloc(sizeof(struct node));
        printf("Enter the data: ");
        scanf("%d",&new->data);
        new->prev = NULL;
        new->next = NULL;
        if(head == NULL){
            head = temp = new;
        }
        else{
            temp->next = new;
            new->prev = temp;
            temp = new;
        }
        printf("Do you want to add another element (0/1): ");
        scanf("%d",&choice);
    }
    return head;
}

struct node* deleteNode(struct node* head){
    int ele,pos;
    struct node *temp = head,*ptr = NULL;
    printf("\nEnter your element to be deleted: ");
    scanf("%d",&ele);
    while(temp->data != ele){
        ptr = temp;
        temp = temp->next;
    }
    if(temp == head){
        head = head->next;
        head->prev = NULL;
    }
    else if(temp->next == NULL){
        ptr->next = NULL;
    }
    else{
        ptr->next = temp->next;
        temp->next->prev = ptr;
    }
    free(temp);
    temp = NULL;
    return head;
}

void display(struct node *head){
    struct node* temp = head;
    while(temp != NULL){
        printf("%d ",temp->data);
        temp = temp->next;
    }
    printf("\n");
}
```

OUTPUT :-

```
Enter the data: 21
Do you want to add another element (0/1): 1
Enter the data: 22
Do you want to add another element (0/1): 1
Enter the data: 23
```

```
Do you want to add another element (0/1): 1
Enter the data: 24
Do you want to add another element (0/1): 1
Enter the data: 25
Do you want to add another element (0/1): 1
Enter the data: 26
Do you want to add another element (0/1): 0
After creation list is:
21 22 23 24 25 26

1.Delete A Node
2.Exit
Enter you choice: 1
Enter your element to be deleted: 21
After deletion list is:
22 23 24 25 26

1.Delete A Node
2.Exit
Enter you choice: 1
Enter your element to be deleted: 26
After deletion list is:
22 23 24 25

1.Delete A Node
2.Exit
Enter you choice: 1
Enter your element to be deleted: 23
After deletion list is:
22 24 25

1.Delete A Node
2.Exit
Enter you choice: 2
```

3. Write a C program implement the Stack ADT using Arrays and Linked List.

Using Array

```c
#include<stdio.h>
#define N 10

int stack[10];
int top = -1;
void push();
void pop();
void peak();
void display();

int main(){
    int choice = 1,option;
    while(choice){
        printf("\n1.Push\n2.Pop\n3.Peak Element\n4.Display\n5.Exit");
        printf("\nEnter your choice: ");
        scanf("%d",&option);
        switch(option){
            case 1:
                push();
                break;
            case 2:
                pop();
                break;
            case 3:
                peak();
                break;
            case 4:
                display();
                break;
            case 5:
                choice = 0;
                break;
        }
    }
    return 0;
}

void push(){
    if(top == N-1){
        printf("Stack is full");
    }
    else{
        int data;
        printf("Enter the data: ");
        scanf("%d",&data);
        stack[++top] = data;
    }
}

void pop(){
    if(top == -1){
        printf("Stack is empty");
    }
    else{
        printf("%d is popped out",stack[top--]);
    }
}

void peak(){
    if(top == -1){
        printf("Stack is empty");
    }
    else{
        printf("%d is peak element",stack[top]);
    }
}

void display(){
    if(top == -1){
        printf("Stack is empty");
    }
    else{
        for(int i=top;i>-1;i--){
            printf("%d ",stack[i]);
        }
    }
}
```

OUTPUT :-

```
1.Push
2.Pop
3.Peak Element
4.Display
5.Exit
Enter your choice: 1
Enter the data: 21
1.Push
2.Pop
3.Peak Element
4.Display
5.Exit
Enter your choice: 1
Enter the data: 26
1.Push
2.Pop
3.Peak Element
4.Display
5.Exit
Enter your choice: 1

3 is popped out
1.Push
2.Pop
3.Peak Element
4.Display
```

```
5.Exit
Enter your choice: 3
26 is peak element

1.Push
2.Pop
3.Peak Element
4.Display
5.Exit
Enter your choice: 4
26 21
1.Push
2.Pop
3.Peak Element
4.Display
5.Exit
Enter your choice: 5
```

Using Linked List

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

struct node{
    int data;
    struct node* link;
}*top = NULL;

void push();
void pop();
void peak();
void display();

int main(){
    int choice = 1,option;
    while(choice){
        printf("\n1.Push\n2.Pop\n3.Peak Element\n4.Display\n5.Exit");
        printf("\nEnter your choice: ");
        scanf("%d",&option);
        switch(option){
            case 1:
                push();
                break;
            case 2:
                pop();
                break;
            case 3:
                peak();
                break;
            case 4:
                display();
                break;
            case 5:
                choice = 0;
                break;
        }
    }
    return 0;
}

void push(){
    struct node *new = (struct node*)malloc(sizeof(struct node));
    printf("Enter the data: ");
    scanf("%d",&new->data);
    new->link = NULL;
    if(top == NULL)
        top = new;
    else{
        new->link = top;
        top = new;
    }
```

```c
}

void pop(){
    if(top == NULL){
        printf("Stack is empty");
    }
    else{
        struct node* temp = top;
        top = top->link;
        printf("%d is popped out",temp->data);
        free(temp);
        temp = NULL;
    }
}

void peak(){
    if(top == NULL){
        printf("Stack is empty");
    }
    else{
        printf("%d is peak element",top->data);
    }
}

void display(){
    if(top == NULL){
        printf("Stack is empty");
    }
    else{
        struct node* temp = top;
        while(temp != NULL){
            printf("%d ",temp->data);
            temp = temp->link;
        }
    }
}
```

OUTPUT :-

```
1.Push
2.Pop
3.Peak Element
4.Display
5.Exit
Enter your choice: 1
Enter the data: 21
1.Push
2.Pop
3.Peak Element
4.Display
5.Exit
Enter your choice: 1
Enter the data: 26
1.Push
```

```
2.Pop
3.Peak Element
4.Display
5.Exit
Enter your choice: 1

3 is popped out
1.Push
2.Pop
3.Peak Element
4.Display
5.Exit
Enter your choice: 3
26 is peak element

1.Push
2.Pop
3.Peak Element
4.Display
5.Exit
Enter your choice: 4
26 21
1.Push
2.Pop
3.Peak Element
4.Display
5.Exit
Enter your choice: 5
```

4. Write a C program that uses stack operations to convert a given infix expression into its postfix equivalent.

```c
#include<stdio.h>
#include<ctype.h>
#define N 20

char stack[20];
int top = -1;

void push(char);
char pop();
int priority(char);

int main(){
    char infix[30],postfix[30];
    char ch;
    int i=0,j=0;

    printf("Enter the infix expression: ");
    gets(infix);

    for(i = 0;infix[i] != '\0';i++){
        ch = infix[i];
        if(ch == '('){
            push(ch);
        }
        else if(isalpha(ch) || isdigit(ch)){
            postfix[j++] = ch;
        }
        else if(ch == ')'){
            while(stack[top] != '(' && top != -1){
                postfix[j++] = pop();
            }
            pop();
        }
        else{
            while(priority(ch)<=priority(stack[top]) && top != -1){
                postfix[j++] = pop();
            }
            push(ch);
        }
    }
    while(top != -1){
        postfix[j++] = pop();
    }
    for(i = 0;i<j;i++){
        printf("%c",postfix[i]);
    }
}

void push(char x){
    stack[++top] = x;
```

```c
}

char pop(){
    return stack[top--];
}

int priority(char x){
    if(x == '(')
        return 0;
    else if(x == '+' || x == '-')
        return 1;
    else if(x == '*' || x == '/')
        return 2;
    else
        return 3;
}
```

OUTPUT :-

```
Enter the infix expression: 2*3/(2-1)+5*3
23*21-/53*+
```

5. Write a C program that evaluates a postfix expression

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

#define N 10
int stack[N];
int top = -1;
void push(int);
int pop();
int main(){
    char exp[30];
    int a,b,d;
    printf("Enter the postfix expression: ");
    scanf("%s",exp);
    for(int i=0;exp[i]!='\0';i++){
        if(isdigit(exp[i])){
            push(exp[i]-'0');
        }
        else{
            a = pop();
            b = pop();
            switch(exp[i]){
                case '+':
                    push(b+a);
                    break;
                case '-':
                    push(b-a);
                    break;
                case '*':
                    push(b*a);
                    break;
                case '/':
                    push(b/a);
                    break;
                case '^':
                    pow(b,a);
                    push(d);
                    break;
            }
        }
    }
    printf("The result is: %d\n",pop());

}
void push(int data){
    top++;
    stack[top] = data;
}
int pop(){
    return stack[top--];
}
```

OUTPUT :-

```
Enter the postfix expression: 2536+**5/2-
The result is: 16
```

6. Write C program to implement queue ADT using array and doubly linked list

Using Array

```c
#include<stdio.h>
#define MAX 4

int queue[MAX];
int front = -1,rear = -1;
void enqueue();
void dequeue();
void peek();
void display();

int main(){
    int choice = 1,option;
    while(choice){
        printf("\n1.Enqueue");
        printf("\n2.Dequeue");
        printf("\n3.Display");
        printf("\n4.Peek");
        printf("\n5.Exit");
        printf("\nSelect your option: ");
        scanf("%d",&option);
        switch(option){
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                peek();
                break;
            case 5:
                choice = 0;
                break;
            default:
                printf("You Entered the wrong choice");
                break;
        }
    }
}

void enqueue(){
    if(rear == MAX - 1)
        printf("Queue Overflow");
    else{
        if(front == -1)
```

```
            front = 0;
        rear = rear + 1;
        printf("Enter the data: ");
        scanf("%d",&queue[rear]);
    }
}

void dequeue(){
    if(front == -1)
        printf("Queue Underflow");
    else if(front == rear){
        printf("%d is deleted",queue[front]);
        front = rear = -1;
    }
    else{
        printf("%d is deleted",queue[front]);
        front = front + 1;
    }
}

void peek(){
    if(rear == -1){
        printf("Queue Underflow");
        return;
    }
    printf("The first element is %d\n",queue[front]);
}

void display(){
    if(rear == -1){
        printf("Queue Underflow");
        return;
    }
    int i;
    for(i=front;i<=rear;i++){
        printf("%d ",queue[i]);
    }
}
```

OUTPUT :-

```
1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 21

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
```

```
Select your option: 1
Enter the data: 22

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 1

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 90

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 2
21 is deleted

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 3
22 1 90

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 4
The first element is 22

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 5
```

Using Double Linked List

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

struct node{
    int data;
    struct node* prev;
    struct node* next;
}*front = NULL,*rear = NULL;

void enqueue();
void dequeue();
void peek();
void display();

int main(){
    int choice = 1,option;
    while(choice){
        printf("\n1.Enqueue");
        printf("\n2.Dequeue");
        printf("\n3.Display");
        printf("\n4.Peek");
        printf("\n5.Exit");
        printf("\nSelect your option: ");
        scanf("%d",&option);
        switch(option){
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                peek();
                break;
            case 5:
                choice = 0;
                break;
            default:
                printf("You Entered the wrong choice");
                break;
        }
    }
}

void enqueue(){
    struct node *new = (struct node*)malloc(sizeof(struct node));
    printf("Enter the data: ");
    scanf("%d",&new->data);
    new->prev = NULL;
    new->next = NULL;
    if(front == NULL){
        front = rear = new;
    }
    else{
        rear->next = new;
        new->prev = rear;
        rear = new;
    }
}

void dequeue(){
    if(front == NULL){
        printf("Queue Underflow\n");
    }
    else{
        struct node *temp = front;
        front = front->next;
        front->prev = NULL;
        if(front == NULL)
            rear = NULL;
        printf("%d is deleted\n",temp->data);
        free(temp);
        temp = NULL;
    }
}

void peek(){
    if(front == NULL){
        printf("Queue Underflow\n");
        return;
    }
    else{
        printf("The first element is %d\n",front->data);
    }
}

void display(){
    if(front == NULL){
        printf("Queue Underflow\n");
        return;
    }
    else{
        struct node *temp = front;
        while(temp != NULL){
            printf("%d ",temp->data);
            temp = temp->next;
        }
    }
}
```

OUTPUT :-

```
1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 21

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 22

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 1

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 90

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 2
21 is deleted

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 3
22 1 90

1.Enqueue
2.Dequeue
```

```
3.Display
4.Peek
5.Exit
Select your option: 4
The first element is 22

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 5
```

7. a) Write C program to implement priority queue ADT using array.
b) Write C program to implement circular queue ADT using array.

Priority Queue

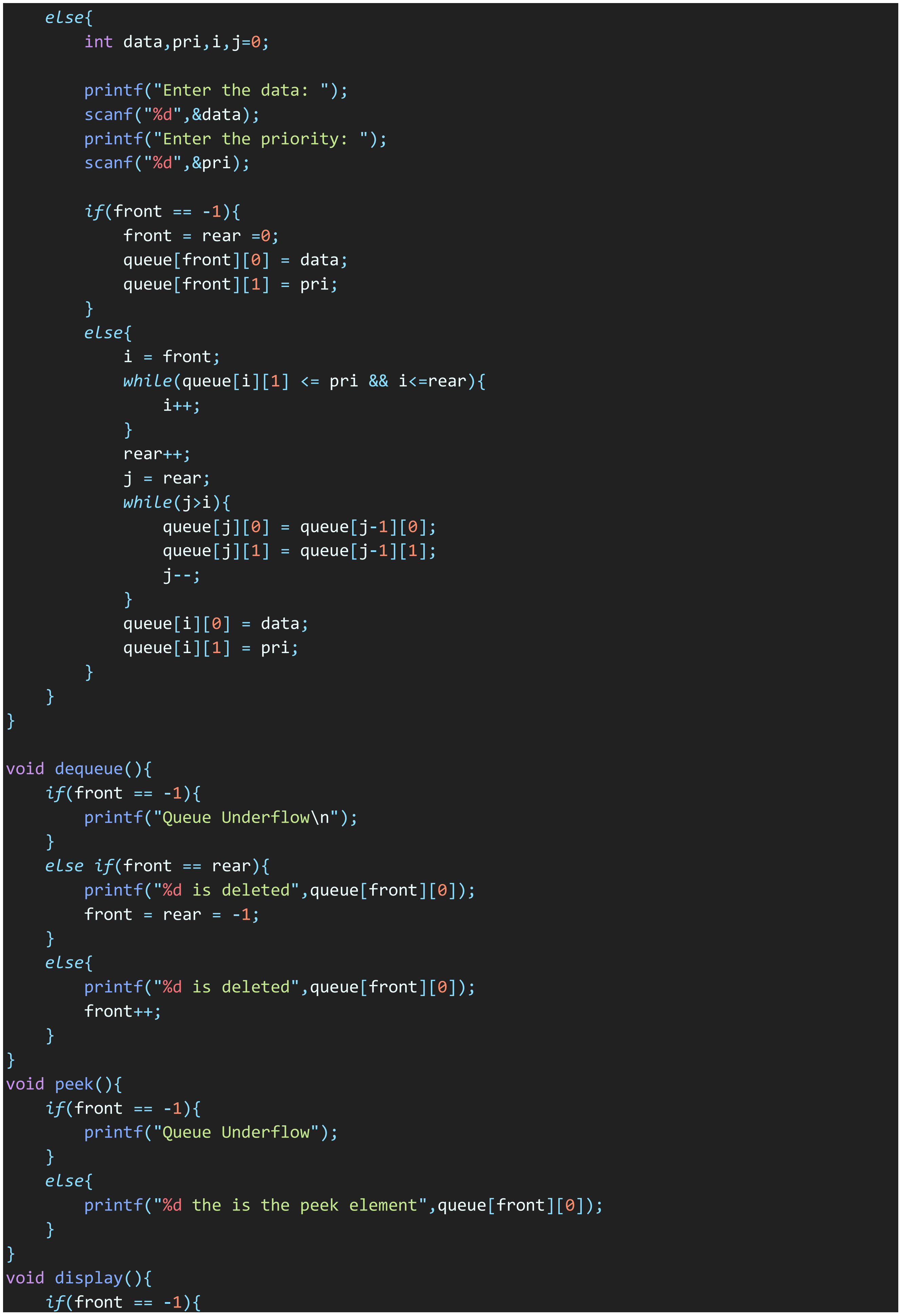
```c
#include<stdio.h>
#define N 10

int queue[10][2], front=-1, rear=-1;

void enqueue();
void dequeue();
void display();
void peek();

int main(){
    int choice = 1,option;
    while(choice){
        printf("\n1.Enqueue");
        printf("\n2.Dequeue");
        printf("\n3.Display");
        printf("\n4.Peek");
        printf("\n5.Exit");
        printf("\nSelect your option: ");
        scanf("%d",&option);
        switch(option){
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                peek();
                break;
            case 5:
                choice = 0;
                break;
            default:
                printf("You Entered the wrong choice");
                break;
        }
    }
    return 0;
}

void enqueue(){
    if(rear == N-1){
        printf("Queue Overflow");
    }
```



```
    else{
        int data,pri,i,j=0;

        printf("Enter the data: ");
        scanf("%d",&data);
        printf("Enter the priority: ");
        scanf("%d",&pri);

        if(front == -1){
            front = rear =0;
            queue[front][0] = data;
            queue[front][1] = pri;
        }
        else{
            i = front;
            while(queue[i][1] <= pri && i<=rear){
                i++;
            }
            rear++;
            j = rear;
            while(j>i){
                queue[j][0] = queue[j-1][0];
                queue[j][1] = queue[j-1][1];
                j--;
            }
            queue[i][0] = data;
            queue[i][1] = pri;
        }
    }
}

void dequeue(){
    if(front == -1){
        printf("Queue Underflow\n");
    }
    else if(front == rear){
        printf("%d is deleted",queue[front][0]);
        front = rear = -1;
    }
    else{
        printf("%d is deleted",queue[front][0]);
        front++;
    }
}
void peek(){
    if(front == -1){
        printf("Queue Underflow");
    }
    else{
        printf("%d the is the peek element",queue[front][0]);
    }
}
void display(){
    if(front == -1){
```

```c
        printf("Queue Underflow\n");
    }
    else{
        for(int i=front;i<=rear;i++){
            printf("%d ",queue[i][0]);
        }
    }
}
```

OUTPUT :-

```
1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 21
Enter the priority: 3

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 16
Enter the priority: 2

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 18
Enter the priority: 2

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 11
Enter the priority: 1

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 3
11 16 18 21

1.Enqueue
2.Dequeue
```

```
3.Display
4.Peek
5.Exit
Select your option: 2
11 is deleted

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 4
16 is the peek element

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 5
```

Circular Queue

```c
#include<stdio.h>
#define N 5
int queue[N];
int front = -1,rear = -1;
void enqueue();
void dequeue();
void peek();
void display();

int main(){
    int choice = 1,option;
    while(choice){
        printf("\n1.Enqueue");
        printf("\n2.Dequeue");
        printf("\n3.Display");
        printf("\n4.Peek");
        printf("\n5.Exit");
        printf("\nSelect your option: ");
        scanf("%d",&option);
        switch(option){
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                peek();
                break;
            case 5:
                choice = 0;
                break;
            default:
                printf("You Entered the wrong choice");
                break;
        }
    }
}

void enqueue(){
    if(front == (rear+1)%N){
        printf("Queue Overflow\n");
        return;
    }
    else{
        int x;
        printf("Enter the data: ");
        scanf("%d",&x);
        if(front == -1 && rear == -1){
            front = rear = 0;
            queue[rear] = x;
        }
        else{
            rear = (rear+1)%N;
            queue[rear] = x;
        }
    }
}

void dequeue(){
    if(front == -1 && rear == -1){
        printf("Queue Underflow\n");
    }
    else if(front == rear){
        printf("%d is deleted\n",queue[front]);
        front = rear = -1;
    }
    else{
        printf("%d is deleted\n",queue[front]);
        front = (front+1)%N;
    }
}

void display(){
    if(front == -1 && rear == -1){
        printf("Queue Underflow\n");
        return;
    }
    int i = front;
    while(i!=rear){
        printf("%d ",queue[i]);
        i = (i+1)%N;
    }
    printf("%d",queue[i]);
}

void peek(){
    if(front == -1 && rear == -1){
        printf("Queue Underflow\n");
        return;
    }
    printf("The first element is %d",queue[front]);
}
```

OUTPUT :-

```
1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 22

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 88

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 11

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 8

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 10

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 2
22 is deleted

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 1
Enter the data: 94
```

```
1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 3
88 11 8 10 94

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 4
The first element is 88

1.Enqueue
2.Dequeue
3.Display
4.Peek
5.Exit
Select your option: 5
```

8. Write C program for implementing the following sorting methods: b) Insertion sort b) Merge sort

Insertion Sort

```c
#include<stdio.h>

int main(){
    int arr[20],n,i,j,min;
    printf("Enter the no of elements: ");
    scanf("%d",&n);
    printf("Enter the array elements:\n");
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }

    for(i=1;i<n;i++){
        min = arr[i];
        j = i-1;

        while(j>=0 && arr[j]>min){
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = min;
    }
    for(i = 0;i<n;i++){
        printf("%d ",arr[i]);
    }
}
```

OUTPUT :-

```
Enter the no of elements: 7
Enter the array elements:
23
1
8
66
9
5
13
1 5 8 9 13 23 66
```

Merge Sort

```c
#include<stdio.h>

void merge(int[],int,int,int);
void mergeSort(int[], int,int);

int main(){
    int n;
    int arr[30];
    printf("Enter the elements: ");
    scanf("%d",&n);
    printf("Enter elements of the array:\n");
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    mergeSort(arr,0,n-1);
    for(int i=0;i<n;i++){
        printf("%d ",arr[i]);
    }
    return 0;
}

void mergeSort(int a[],int lb,int ub){
    int mid;
    if(lb<ub){
        mid=(ub+lb)/2;
        mergeSort(a,lb,mid);
        mergeSort(a,mid+1,ub);
        merge(a,lb, mid,ub);
    }
}

void merge(int a[],int lb,int mid,int ub){
    int b[30],i,j,k;
    i=lb;
    j=mid+1;
    k=0;

    while(i<=mid && j<=ub){
        if(a[i]<=a[j]){
            b[k]=a[i];
            i++;
        }
        else{
            b[k]=a[j];
            j++;
        }
        k++;
    }
    while(j<=ub){
        b[k]=a[j];
        j++;
        k++;
    }
    while(i<=mid){
        b[k]=a[i];
        i++;
        k++;
    }
    i=0;
    for(k=lb;k<=ub;k++){
        a[k]=b[i];
        i++;
    }
}
```

OUTPUT :-

```
Enter the no of elements: 7
Enter the array elements:
23
1
8
66
9
5
13
1 5 8 9 13 23 66
```

9. Write C program for implementing the following sorting methods: b) Quick sort b) Selection sort

Quick Sort

```
#include <stdio.h>
void swap(int* ,int*);
int partition(int[],int,int);
void quickSort(int[],int,int);

int main() {
  int a[30];
  int n;
  printf("Enter the no of elements of the array: ");
  scanf("%d",&n);
  printf("Enter the elements of the array:\n");
  for(int i=0;i<n;i++){
    scanf("%d",&a[i]);
  }

  quickSort(a, 0, n - 1);

  printf("Sorted array:\n");
  for(int i=0;i<n;i++){
    printf("%d ",a[i]);
  }
}

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

int partition(int arr[], int lb, int ub) {

  int pivot = lb;
  int i = lb;
  int j = ub;
  while(i<j){
    while(arr[i]<=arr[pivot] && i<ub){
      i++;
    }
    while(arr[j]>arr[pivot]){
      j--;
    }
    if(i<j){
      swap(&arr[i],&arr[j]);
    }
  }
  swap(&arr[j],&arr[pivot]);
  return j;
}
```

```
void quickSort(int arr[], int lb , int ub) {
  if (lb < ub) {
    int p = partition(arr, lb, ub);
    quickSort(arr, lb, p-1);
    quickSort(arr, p+1, ub);
  }
}
```

OUTPUT :-

```
Enter the no of elements: 7
Enter the array elements:
23
1
8
66
9
5
13
1 5 8 9 13 23 66
```

Selection Sort

```c
#include<stdio.h>
int main(){
    int arr[30];
    int n,min,temp;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    printf("Enter the array elements: \n");
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    for(int i=0;i<n-1;i++){
        min = i;
        for(int j=i+1;j<n;j++){
            if(arr[min]>arr[j])
                min = j;
        }
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
    printf("The sorted elements are: ");
    for(int i=0;i<n;i++){
        printf("%d ",arr[i]);
    }

}
```

OUTPUT :-

```
Enter the no of elements: 7
Enter the array elements:
23
1
8
66
9
5
13
1 5 8 9 13 23 66
```

10. Write a C program for implementing Heap sort algorithm.

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

void heapSort(int[], int);
void maxHeapify(int[], int, int);
void swap(int*, int*);

int main(){
    int a[30];
    int n;
    printf("Enter the no of elements of the array: ");
    scanf("%d",&n);
    printf("Enter the elements of the array:\n");
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }

    heapSort(a,n);

    printf("Sorted array:\n");
    for(int i=0;i<n;i++){
        printf("%d ",a[i]);
    }
}

void heapSort(int a[], int n){
    for(int i = n/2-1;i>=0;i--){
        maxHeapify(a,n,i);
    }
    for(int i = n-1;i>0;i--){
        swap(&a[0],&a[i]);
        maxHeapify(a,i,0);
    }
}

void maxHeapify(int a[], int n, int i){
    int largest = i;

    int left = 2*i + 1;
    int right = 2*i + 2;

    if(left < n && a[left] > a[largest]){
        largest = left;
    }

    if(right < n && a[right] > a[largest]){
        largest = right;
    }

    if(largest != i){
        swap(&a[largest],&a[i]);
        maxHeapify(a,n,largest);
    }
}

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

OUTPUT :-

```
Enter the no of elements: 7
Enter the array elements:
23
1
8
66
9
5
13
1 5 8 9 13 23 66
```

11. Write a C program that uses functions to perform the following:
a) Create a Binary Search Tree (BST).
b) Insert data in BST
c) Traverse the above BST recursively in Postorder

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

struct node{
    struct node* left;
    int data;
    struct node* right;
};

struct node* createNode(int);
struct node* createBST(struct node*);
struct node* insertNode(struct node*,int);
struct node* insertElement(struct node*, int);
void printBST(struct node*);

int main(){
    struct node* root = NULL;
    root = createBST(root);
    printBST(root);
    int data;
    printf("\nEnter the element you want to insert: ");
    scanf("%d",&data);
    root = insertElement(root,data);
    printBST(root);
}

struct node* createNode(int data){
    struct node* ptr = (struct node*)malloc(sizeof(struct node));
    ptr->data = data;
    ptr->left = NULL;
    ptr->right = NULL;

    return ptr;
}

struct node* createBST(struct node* root){
    int data;
    int choice = 1;
    while(choice){
        printf("Enter the data: ");
        scanf("%d",&data);
        root = insertNode(root,data);
        printf("Do you want to continue (1-Yes/0-No): ");
        scanf("%d",&choice);
    }
    return root;
}
```

```c
struct node* insertNode(struct node* root, int data){
    if(root == NULL){
        root = createNode(data);
        return root;
    }
    else{
        if(data < root->data)
            root->left = insertNode(root->left,data);
        else if(data > root->data)
            root->right = insertNode(root->right,data);
        return root;
    }
}

void printBST(struct node* root){
    if(root == NULL)
        return;
    else{
        printBST(root->left);
        printBST(root->right);
        printf("%d ",root->data);
    }
}

struct node* insertElement(struct node* root, int data){
    if(root == NULL)
    {
        root = createNode(data);
    }
    else if(data < root->data)
    {
        root->left = insertElement(root->left,data);
    }
    else if(data > root->data)
    {
        root->right = insertElement(root->right,data);
    }
    else
    {
        printf("Duplicate Key!");
    }
    return root;
}
```

OUTPUT :-

```
Enter the data: 22
Do you want to continue (1-Yes/0-No): 1
Enter the data: 28
Do you want to continue (1-Yes/0-No): 1
Enter the data: 88
Do you want to continue (1-Yes/0-No): 1
Enter the data: 16
Do you want to continue (1-Yes/0-No): 1
Enter the data: 7
```

```
Do you want to continue (1-Yes/0-No): 0
7 16 88 28 22
Enter the element you want to insert: 35
7 16 35 88 28 22
```

12. Write a C program that uses functions to perform the following:
a) Deletion an element BST
b) Traverse the above BST non recursively in Inorder

```c
#include<stdio.h>
#include<stdlib.h>
#define MAX 20
struct node{
    struct node* left;
    int data;
    struct node* right;
};
struct node* stack[MAX];
int top = -1;

void push(struct node*);
struct node* pop();
int isEmpty();
struct node* createNode(int);
struct node* createBST(struct node*);
struct node* insertNode(struct node*,int);
struct node* delNode(struct node*,int);
struct node* delcaseOne(struct node*, struct node*, struct node*);
struct node* delcaseTwo(struct node*, struct node*, struct node*);
struct node* delcaseThree(struct node*, struct node*);
void printBST(struct node*);

int main()
{
    struct node* root = NULL;
    int data;
    root = createBST(root);
    printBST(root);
    printf("\nEnter the data you want to delete: ");
    scanf("%d",&data);
    root = delNode(root,data);
    printf("\nAfter deletion:\n");
    printBST(root);
    return 0;
}

struct node* createNode(int data){
    struct node* ptr = (struct node*)malloc(sizeof(struct node));
    ptr->data = data;
    ptr->left = NULL;
    ptr->right = NULL;

    return ptr;
}

struct node* createBST(struct node* root){
    int data;
    int choice = 1;
    while(choice){
        printf("Enter the data: ");
        scanf("%d",&data);
        root = insertNode(root,data);
        printf("Do you want to continue (1-Yes/0-No): ");
        scanf("%d",&choice);
    }
    return root;
}

struct node* insertNode(struct node* root, int data){
    if(root == NULL){
        root = createNode(data);
        return root;
    }
    else{
        if(data < root->data)
            root->left = insertNode(root->left,data);
        else if(data > root->data)
            root->right = insertNode(root->right,data);
        return root;
    }
}

void printBST(struct node* root){
    struct node* ptr = root;
    if(ptr == NULL){
        printf("Tree is empty\n");
    }
    while(1){
        while(ptr->left != NULL){
            push(ptr);
            ptr = ptr->left;
        }
        while(ptr->right == NULL){
            printf("%d ",ptr->data);
            if(isEmpty())
                return;
            ptr = pop();
        }
        printf("%d ",ptr->data);
        ptr = ptr->right;
    }
}

struct node* delNode(struct node* root, int data)
{
    struct node *ptr = root,*temp = NULL;
    while(ptr != NULL)
    {
        if(data == ptr->data)
            break;
        temp = ptr;
        if(data < ptr->data)
            ptr = ptr->left;
        else
            ptr = ptr->right;
    }
    if(ptr->left == NULL && ptr->right == NULL)
        root = delcaseOne(root,ptr,temp);
    else if(ptr->left == NULL || ptr->right == NULL)
        root = delcaseTwo(root,ptr,temp);
    else
        root = delcaseThree(root,ptr);
    return root;
}

struct node* delcaseOne(struct node* root,struct node* ptr, struct node* temp)
{
    if(ptr == NULL)
        root = NULL;
    else if(ptr == temp->left)
        temp->left = NULL;
    else
        temp->right = NULL;
    free(ptr);
    ptr = NULL;
    return root;
}

struct node* delcaseTwo(struct node* root, struct node* ptr, struct node* temp)
{
    struct node* child = NULL;

    if(ptr->left == NULL)
        child = ptr->right;
    else
        child = ptr->left;

    if(temp->left == NULL)
        temp->right = child;
    else
        temp->left = child;
    free(ptr);
    ptr = NULL;
    return root;
}

struct node* delcaseThree(struct node* root, struct node* ptr)
{
    struct node* temp = ptr;
    struct node* successor = ptr->right;

    while(successor->left != NULL)
    {
        temp = successor;
        successor = successor->left;
    }
    ptr->data = successor->data;
    if(successor->left == NULL && successor->right == NULL)
        root = delcaseOne(root, successor, temp);
    else
        root = delcaseTwo(root, successor, temp);
    return root;
}

void push(struct node* ptr){
    stack[++top] = ptr;
}

struct node* pop(){
    return stack[top--];
}

int isEmpty(){
    if(top == -1)
        return 1;
    else
        return 0;
}
```

OUTPUT :-

```
Enter the data: 21
Do you want to continue (1-Yes/0-No): 1
Enter the data: 18
Do you want to continue (1-Yes/0-No): 1
Enter the data: 15
Do you want to continue (1-Yes/0-No): 1
Enter the data: 12
Do you want to continue (1-Yes/0-No): 1
Enter the data: 87
Do you want to continue (1-Yes/0-No): 1
Enter the data: 66
Do you want to continue (1-Yes/0-No): 0
12 15 18 21 66 87
Enter the data you want to delete: 66
After deletion:
12 15 18 21 87
```