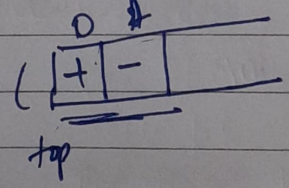
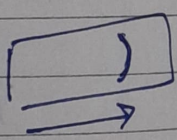
```
int pred(char symbol){
    int p;
    switch(symbol){
        case '^' : p=3;
                   break;
        case '*':
        case '/' : p=2;
                   break;
        case '+':
        case '-' : p=1;
                   break;
        case '(' : p=0;
                   break;
        case '#' : p=-1
                   break;
    }
    return p;
```

Evaluation 3
4/1/2024.





A+B*(C+D)/F+D*E)

Stack

(

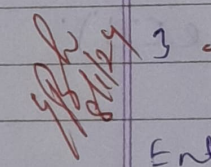
(+*(+)/

*

(+*(+

ABCD+*F/+DE

```
        scanf("%d", &n);
        push(n);
        break;
case 2: n = pop();
        printf("%d is poped", n);
        break;
case 3: printf("queue is \n");
        display();
        break;
default: printf("invalid choice\n
        break;
}
}
}
}
```



Enter 1Enqueue
2. Dequeue
3. Display
4. -1 to stop execution
Enteroperator
2
Queue is empty(underflow
-1 is decued
Enteroperator
1
Enter no
2

```
void display (struct node *head){
    if (head != NULL){
        printf("Elements are :\t");
        while ( head != NULL){
            printf ("%d\t", head -> data);
            head = head -> next;
        }
        printf("\n");
    }
    else {
        printf(" Linked list is empty\n");
    }
}

struct node * delAtStart ( struct node **head){
    if (*head != NULL){
        struct node *temp = *head;
        *head = (*head) -> next;
        temp -> next = NULL;
        return NULL;
    }
}
```

22/11/24

```
struct node * delAtEnd ( struct node **head){
    if (*head != NULL){
        if ((*head) -> next == NULL){
            struct node *temp = *head;
            *head = NULL;
            return temp;
        }
        else {
            struct node *ptr = *head;
            struct node *prev = NULL;
```

Demonstration of Leetcode.

Demonstration shown 29/1/24

— Go.
—
—

created and loged in account.
For solving questions we need to consider the parameters given and return the required answer as the platform wants. Then it checks our solutions with some predefined test cases and displays our stand and codes efficiency amoung the global users.

Date
Page

3

return 0;

}

O/P

Enter the number of nodes to create initially: 3
Enter data for node 1: 1
Enter data for node 2: 2
Enter data for node 3: 3
1. Insert
2. Delete
3. to stop
Doubly Linked List: 1 2 3
Enter your choice: 2
Enter the value to delete: 3
Node with data 3 deleted.
Double linked list: 1 2
Enter your choice: 1
Enter data for the target node: 1
Enter data for new node: 0
Node with data 0 inserted to the left of node with data 1.
Doubly linked list: 0 1 2
Enter choice: 3

```
int main(){
    struct Node *root=NULL;
    int data, c;
    printf("1. Enter data into BST \n2.
                    To stop\n");
    while(1){
        printf("Enter choice: ");
        scanf("%d", &c);
        switch(c){
            case 1:
                printf("Enter data: ");
                scanf("%d", &data);
                root = insert(root, data);
                Break;
            case 2:
                display(root);
                exit(0);
        }
    }
    return 0;
}
```

BST - Lecture 01

NLP 19/2/24

Output:

```
1. Enter data into BST
2. To stop
Enter choice: 1
Enter data: 2
Enter choice: 1
Enter data: 5
Enter choice: 1
Enter data: 7
Enter choice: 1
```

```
else if (root->right==NULL){
        struct TreeNode *temp=root->left;
        free(root);
        return;
    }
    struct TreeNode *temp=root->right;
    while(temp->left!=NULL){
        temp=temp->left;
    }
    root->val=temp->val;
    root->right=deleteNode(root->right,temp
                                    ->val);
}
return root;
}
```

2) Bottom left tree value.

```
int findBottomLeftValue(struct TreeNode *root){
    if(root==NULL){return -1;}
    struct TreeNode *queue[10000];
    int front=0,rear=0,leftmostValue=-1;
    queue[rear++]=root;
    while(front<rear){
        int levelSize=rear-front;
        for(int i=0;i<levelSize;++i){
            struct TreeNode *node=queue[front++];
            if(i==0){
                leftmostvalue=node->val;
            }
            if(node->left!=NULL){
                queue[rear++]=node->left;
            }
            if(node->right!=NULL){
                queue[rear++]=node->right;
            }
        }
    }
    return leftmostvalue;
}
```

20/2/24