

INDEX

Name : VARUN VINOD

Subject : D.S

Std. : Div. :

Roll No. :

School / College :

Sl No.	Date	Title	Page No.	Teacher Sign/ Remarks
1	11/12/23	array expense program, inch/foot sum program, Student backlog program, library program		<i>Shi</i>
2	18/12/23	Dynamic memory allocation program		<i>Shi</i>
3	11/1/24	Stack / infix to postfix program		<i>Shi</i>
4	8/1/24	Queue, circular queue		<i>Shi</i>
5	27/1/24	Singly linked list (insertion/deletion)		<i>Shi</i>
6	29/1/24	Sorting, concatenate, reverse, Stack/queue implementation		<i>Shi</i>
7	3/2/24	Doubly linked list		<i>Shi</i>
8	19/2/24	Binary tree operation, Delete middle node, odd even linked list leetcode.		<i>Shi</i>
9	26/2/24	BFS, DFS, leetcode problems		<i>Shi</i>

1. insert 2. delete 3. display 4. exit
Enter the choice

1
Enter item to be inserted

5
Successfully inserted!

1. insert 2. delete 3. display 4. exit
Enter the choice

2
Deleted element is 2

~~1. insert 2. delete 3. display~~
~~Enter the choice~~

~~4~~

~~08/10/24~~

1. insert 2. delete 3. display 4. exit

Enter the choice

2

Deleted element is 5.

1. insert 2. delete 3. display

Enter the choice

2

Underflow!

Q/P Linked list: $1 \rightarrow 2 \rightarrow 3 \rightarrow \text{NULL}$
Linked list: $2 \rightarrow 3 \rightarrow \text{NULL}$
Linked list: $3 \rightarrow \text{NULL}$
~~Linked list: NULL~~

~~Ans~~
22/01/24


```

if (front == rear)
{
    printf("Queue is empty\n");
}
else
{
    temp = front;
    while (temp != -1)
    {
        front = front - 1;
        free(temp);
    }
}
}

```

o/p After enqueueing

10 ← 20 ← 30 ← NULL

Dequeued value: 10

Queue after dequeue

20 ← 30 ← NULL

29.01.24

```

    }
    printf("Empty\n");
}
else
    printf("\n");
    int val;
    printf("Enter the value ");
    scanf("%d", &val);
    temp = head;
    while (temp -> data != val)
    {
        temp = temp -> next;
    }
    if (temp -> next -> next == NULL)
    {
        temp -> next = NULL;
        printf("Node deleted");
    }
    else
    {
        ptr = temp -> next;
        temp -> next = ptr -> next;
        ptr -> next -> prev = temp;
        free(ptr);
        printf("Node deleted");
    }
}
}

```

05.02.24

```

struct node* temp = root -> left
while (temp -> right != NULL)
    temp = temp -> right;
root -> data = temp -> data
root -> left = deletenode(root -> left, temp -> data)
}
}
return root;
}

```

→ Find bottom left tree value

```

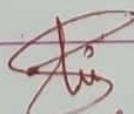
int Bottomval (struct Node* root)
{
    if (root == NULL)
        return -1;

    int leftmostval = root -> data;
    int maxDepth = 0;
    void dfs (struct node*, int depth) {
        if (root == NULL)
            return -1;
        if (depth > maxDepth) {
            leftmostval = node -> data;
            maxDepth = depth;
        }

        dfs (node -> left, depth++);
        dfs (node -> right, depth++);
    }

    dfs (root, 0);
    return leftmostval;
}

```


 26.02.24