# 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 | daily expense program, inch/feet sum program, Student backlog program, library program | | [illegible] |
| 2 | 18/12/23 | Dynamic memory allocation program | | [illegible] |
| 3 | 1/1/24 | Stack / infix to postfix program | | [illegible] |
| 4 | 8/1/24 | Queue, circular queue | | [illegible] |
| 5 | 22/1/24 | Singly linked list (insertion/deletion) | | [illegible] |
| 6 | 29/1/24 | sorting, concatenate, reverse, Stack/queue implementation | | [illegible] |
| 7 | 5/2/24 | Doubly linked list | | [illegible] |
| 8 | 19/2/24 | Binary tree operation, Delete middle node, Odd even linked list leetcode. | | [illegible] |
| 9 | 26/2/24 | BFS, DFS, leetcode problems | | [illegible] |
| | | | | |
| | | | | |
| | | | | |

```
1. insert  2. delete  3. display  4. e
Enter the choice
1
Enter item to be inserted
5
Succesfully inserted!

1. insert  2. delete  3. display  4. e
Enter the choice
2
Deleted Element is 2

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

1. insert  2. delete  3. display  4.
Enter the choice
2
Deleted Element is 5.

1. insert  2. delete  3. display
Enter the choice
2
Underflow!
```

[illegible] 08/01/24

O/P Linked list: 1 —> 2 —> 3 —> NULL
Linked list: 2 —> 3 —> NULL
Linked list: 3 —> NULL
Linked list: NULL

[illegible] 22/01/24

Scanned with CamScanner

```
    if (front
        {
        printf(" Queue empt
        }
    else
        temp=front;
        while (temp!=rear)
            {
            front=front->
            free (temp);
            }
    }
```

```
O/P  After enqueuing
     10 <- 20 <- 30 <- NULL
     Dequed value: 10
     Quee after dequeu
     20 <- 30 <- NULL
```

[illegible] 29.01.24

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

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

     dfs(node -> left, depth++);
     dfs(node -> right, depth +1);
  }
  dfs(root, 0);
  return leftmostval
}
```

[illegible]
26.02.24