```
void display ()
{
    int i ;
    for ( ~~top=~~ i = top ; i >= 0 ; i--)
        {
            printf ("%d", stack[~~top~~]);
        }
}
```

//output

Enter the operation:

1. push
2. pop
3. display
4. -1 to stop

1

Enter the number

7

successfully pushed

Enter the operation:

1. push
2. pop
3. display
4. -1 to stop

-1

operation completed.

```
while (top > 0)
  {
    ~~top~~ temp = pop()
      postfix [pos++] = temp ;
  }
}
```

Entered

```
            for (i=front; i<=rear; i++)
            {
                printf("%d\n", queue[i]);
            }
        }
    }
```

//Output

Enter operation

1. enque
2. dequeue
3. display
4. -1 to stop

o/p

1

Enter the number:

7

succesfully enqueued

Enter the operation:

1. enqueue
2. dequeue
3. display
4. -1 to stop

8

succesfully enqueued

Enter operation

1. enqueue
2. dequeue
3. display
4. -1 to stop

3.

Elem

Date :
Page No :

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

o/p queue

```
void display()
{
    int i; if (front==-1) { printf("queue empty\n"); }
else {printf("Elements are:\n");

    for (i=front; i!=rear; i=(i+1)%N)
    {
        printf("%d\n",queue[i]);
    }
}
}
```

for display

1. WAP to implement singly linked list with following operations: a) create a linked list b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.

Program

```
#include<stdio.h>

struct node{
        int data;
        struct node * next;
};
struct node *head=NULL, *newnode, *temp;
~~int~~
void create()
{
    int i, n;
    printf("Enter the no. of elements:\n");
    scanf("%d", &n);

    for(i=0; i<n; i++)
    {
        newnode=(struct node *)malloc(sizeof
                                (structnode));
        printf("Enter the %d element:\n", i+1);
        scanf("%d", &newnode->data);
        newnode->next=NULL;

        if (head==NULL)
        { temp=head=newnode;
        }
```

4. delete at end
5- delete at position
6. -1 to end
Enter operation : 1
enter the number of elements : 4
Enter the element 1 : 5
Enter the element 2 : 6
Enter the element 3 : 7
Enter the element 4 : 8

Enter operation : 2
The elements are : 5
6
7
8

Enter operation : 3
Enter operation : 2
The elements are : 6
7
8

Enter operation : 4
Enter operation : 2
The elements are : 6
7

Enter operation : 5
Enter the position : 2
Enter operation : 2
The elements are : 6
~~En~~
Enter operation : -1
operation completed !

```
    if(top==0){printf("stack underflow");}
    else
     { printf("poped element is %d\n", top->data);
       top=top->next;
       free(temp);
     }
}
void main()
{ int choice, num;
  printf("Enter the operation:\n1.push\n2.pop\n
          3.display\n");
  while(1){
          printf("Enter operation\n");
          scanf("%d", &choice);
          if(choice==-1)
          { printf("operation completed.\n");
            break;
          }
  else
   { switch(choice)
     {
      case 1: printf("Enter the number:\n");
              scanf("%d", &num);
              push(num);
              break;
      case 2: pop();
              break;
      case 3: display();
              break;
      default: printf("invalid input");
     }
   }
  }
}
```

```
        while (temp != 0)
        { printf("%d", temp->data);
            temp = temp->next;
        }
    }
}
void dequeue()
{ struct node *temp;
    temp = front;
    if (front == 0 && rear == 0)
        { printf("Queue underflow\n"); }
    else
        { printf("dequeued element is %d\n",
                                front->data);
            front = front->next;
            free(temp);
        }
}

void main()
{ int choice, num;
    printf("Enter the operation:\n1. enque\n
            2. deque.\n3. display\4. -1 to stop\n");
    while(1)
    { printf("Enter operation:\n");
        scanf("%d", &choice);
        if (choice == -1)
        { printf("operation completed\n");
            break;
        }
        else
        { switch (choice)
            {
```

SLL - queue stacks

Ent 29/11/24

```
        case 1: printf("Enter the number:\n");
                scanf("%d", &num);
                enqueue(num);
                break;
        case 2: dequeue();
                break;
        case 3: display();
                break;
        default: printf("Invalid input\n");
    }
  }
}
}
```

29/1/24

//output:

enter the operation

1. enque 2. dequeue 3. display 4. -1 to stop

enter operation 1

enter the number 5

enter operation 1

enter the number 6

enter operation 1

enter the number 7

enter operation 3

5

6

7

enter operation 2

dequeued element is 5

enter operation 2

dequeued element is 6

1a. Write a program to sort and reverse a single linked list.

Demonstration
Lab work
29/1/24

1. Write a program to sort, reverse and concatenation of single linked list.

```
#include <stdio.h>

~~struct node *head=0, *newnode, *temp;~~
struct node {
        int data;
        struct node *next;
};
struct node *head=0, *newnode, *temp;

void create()
{ int i, n;
  printf("Enter the no. of elements:\n");
  scanf("%d", &n);

  for (i=0; i<n; i++)
  {
    newnode = (struct node *) malloc(sizeof(struct node));

    printf("Enter the %d element:\n", i+1);
    scanf("%d", &newnode->data);
    newnode->next = 0;
```

```
        case 2: display ()
                break;
        case 3: reverse ()
                break;
        case 4: sort()
                break;
        case 5: printf("Enter first list\n");
                create();
                printf("enter second list\n");
                create();
                break;
        default: printf("invalid input\n");
    }
  }
 }
}
```

//output:

```
Enter operation:
1. create
2. display
3. reverse
4. sort
5. concat
6. -1 to end
Enter the operation: 1
Enter the no. of elements: 3
Enter the 1 element: 5
Enter the 2 element 6
Enter the 3 elent: 7
Enter operation : 3
Enter operation: 2
The elements are: 7
                  6
                  5
```

enter Operation 2
5
6
7
enter operation 3
enter the node 1
enter data 8
enter operation 2
8
5
6
7
enter operation 3
enter the node 3
Enter data 9
enter operation 2
8
5
9
6
7
enter operation
4
enter position 4
enter operation 2
8
5
9
7
enter operation -1
operation completed

leetcode - score of parantheses

~~class Solution~~

Program

```
int scoreOfParantheses (char *s){
        int stack[1000];
        int top=-1;
        stack[++top]=0;

        for(int i=0; s[i]!='\0'; i++)
        {
          if (s[i]=='(')
                stack[++top]=0;
          } else {
                int a = stack[top--];
                if (a==0)
                  } stack[top] = stack[top]+1;
                  }
                else {
                      stack[top]= stack[top]+2a
                  }
            }
        }
      return stack[0];
}
```

//output

input: s = "()"

output: 1

input: s = "(())"

output: 2

AD
19/2/24

```
        if (temp->data > root
        {
            if (root->right != NULL)
                insert(root->right, temp)
            else
                root->right = temp;
        }
    }

void inorder(struct BST *root)
{ if (root != NULL)
    { inorder(root->left);
      printf("%d", root->data);
      inorder(root->right);
    }
}
void postorder(struct BST *root)
{ if (root != NULL)
    {
      postorder(root->left);
      postorder(root->right);
      printf("%d", root->data);
    }
}
void preorder(struct BST *root)
{ if (root != NULL)
    { printf("%d", root->data);
      preorder(root->left);
      preorder(root->right);
    }
}
```

Date :
Page No :

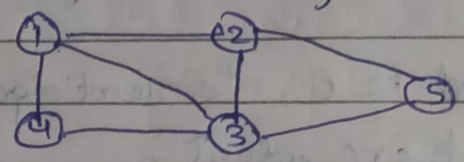
2 leetcode : Delete middlenode of a linkedlist

```
struct ListNode *deleteMiddle(struct ListNode *head)
{
    struct ListNode *temp;
    struct ListNode *midnode = NULL;
    int n = 0, i = 1;
    temp = head;
    while (temp != NULL)
    {
        n++;
        temp = temp->next;
    }
    temp = head;
    int mid = n/2;
    if (head->next == NULL)
    {   head = NULL;
        return head;
    }
    else {
        while (i < mid)
        {
            temp = temp->next;
            i++;
        }
    }
    midnode = temp->next;
    temp->next = midnode->next;
    free (midnode);

    return head;
}
```

Enter edge 1 (source destination): 1 2
Enter edge 2 (source destination): 1 3
Enter edge 3 (source destination): 1 4
Enter edge 4 (source destination): 2 3
Enter edge 5 (source destinatination): 2 5
Enter edge 6 (source destination): 3 4
Enter edge 7 (source destination): 3 5

Enter the starting node for BFS traversal: 1
BFS traversal starting from node 1: 1 4 3 2 5



26/2/24

Leetcode 2: Find Bottom Left Tree value

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

26/2/24

```
    newNode = createNode(src);
    newNode->next = graph->adjlists[dest];
    graph->adjlists[dest] = newNode;
}
void DFS(struct Graph *graph, int startNode){
    graph->visited[startNode]=1;
    printf("%d", startNode);

    struct node * temp = graph->adjlists[startNode];
    while(temp){
        int adjNode = temp->data;
        if(!graph->visited[adjNode]){
            DFS(graph, adjNode);
        }
        temp = temp->next;
    }
}
int main(){
    int numNodes;
    printf("Enter the number of nodes:");
    scanf("%d", &numNodes);

    struct Graph *graph = createGraph(numNodes);
    int numEdges;
    printf("Enter the number of edges:");
    scanf("%d", &numEdges);

    for(int i=0; i<numEdges; i++){
        int src, dest;
        printf("Enter edge %d (src, dest):", i+1);
        scanf("%d %d", &src, &dest);
        addEdge(graph, src, dest);
    }
```

leetcode 1: Delete Node in a BST

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

        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;
}
```

26/2/24