Q. Write a C Program to implement Stack Operations Using Array.

```
#include <stdio.h>
int stack[100],i,j,choice=0,n,top=-1;
void push(); void pop(); void show();
void main ()
{

 while(choice != 4)
{
printf("Chose one from the below options...\n");
printf("1.Push\n2.Pop\n3.Show");
printf("\n Enter your choice \n"); scanf("%d",&choice);
switch(choice)
{
case 1:
push(); break;
case 2:
pop(); break;
case 3:
show(); break;
default:
{
printf("Please Enter valid choice ");
}
}
}
}

void push ()
{
int val;
if (top == n ) printf("\n Overflow");
else
{
printf("Enter the value?"); scanf("%d",&val);
top = top +1; stack[top] = val;
}
}

void pop ()
{
if(top == -1) printf("Underflow");
else
```

```
top = top -1;
}
void show()
{
for (i=top;i>=0;i--)
{
printf("%d\n",stack[i]);
}
if(top == -1)
{
printf("Stack is empty");
}
}
```

/
*** C program to check sparse matrix**
***/**

```
#include <stdio.h>
#define SIZE 3

int main()
{
    int A[SIZE][SIZE];
    int row, col, total=0;

    /* Input elements in matrix from user */
    printf("Enter elements in matrix of size 3x3: \n");
    for(row=0; row<SIZE; row++)
    {
        for(col=0; col<SIZE; col++)
        {
            scanf("%d", &A[row][col]);
        }
    }

    /* Count total number of zero elements in the matrix */
    for(row=0; row<SIZE; row++)
```

```
    {
        for(col=0; col<SIZE; col++)
        {
            /* If the current element is zero */
            if(A[row][col] == 0)
            {
                total++;
            }
        }
    }

    if(total >= (row * col)/2)
    {
        printf("\nThe given matrix is a Sparse matrix.");
    }
    else
    {
        printf("\nThe given matrix is not Sparse matrix.");
    }

    return 0;
}
```

WAP addition of two matrix using 2d array.

```
#include <stdio.h> int main()
 {

int a[2][3], b[2][3], c[2][3], i, j; printf("enter first matrix 6 element: \n"); for(i=0; i<2; i++)
 {

for(j=0; j<3; j++) {

scanf("%d",&a[i][j]); printf("%d\t",a[i][j]);

}

printf("\n");

}
 printf("enter Second matrix 6 element: \n"); for(i=0; i<2; i++)
 {
```

```
for(j=0; j<3; j++) {

scanf("%d",&b[i][j]);

printf("%d\t",b[i][j]); }

printf("\n"); }

printf("addition of two matrix: \n"); for(i=0; i<2; i++)
 {

for(j=0; j<3; j++) {

c[i][j]=a[i][j]+b[i][j]; //scanf("%d",&b[i][j]); printf("%d\t",c[i][j]);

}

printf("\n"); }

return 0; }
```

Q. Write a C Program to implement Queue Operations Using Arrays.

```
#include<stdio.h>
#include<stdlib.h>
#define maxsize 5
void insert();
void delete(); void display();
int front = -1, rear = -1; int queue[maxsize];
```

```
void main ()
{
int choice; while(choice != 4)
{

printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n");
printf("\nEnter your choice ?");
scanf("%d",&choice);
switch(choice)
{
case 1:
insert(); break; case 2:
delete(); break; case 3:
display(); break;
default:
printf("\nEnter valid choice??\n");
}
}
}
void insert()
{
int item;
printf("\nEnter the element\n"); scanf("\n%d",&item);
if(rear == maxsize-1)
{
printf("\nOVERFLOW\n"); return;
}
if(front == -1 && rear == -1)
{
front = 0;
rear = 0;
}
else
{
rear = rear+1;
}
queue[rear] = item; printf("\nValue inserted ");

}
void delete()
{
int item;
if (front == -1 || front > rear)
{
```

```
printf("\nUNDERFLOW\n"); return;

}
else
{
item = queue[front]; if(front == rear)
{
front = -1; rear = -1 ;
}
else
{
front = front + 1;
}
printf("\nvalue deleted ");
}
}

void display()
{
int i;
if(rear == -1)
{
printf("\nEmpty queue\n");
}
else
{ printf("\nprinting values        \n");
for(i=front;i<=rear;i++)
{
printf("\n%d\n",queue[i]);
}
}
}
```

WAP multiplication of two matrix using 2d array.

```
.#include <stdio.h>

#include <stdlib.h>

int main()
```

```
{
    int sum = 0; int i,j,k;
    int a[3][4], b[4][2], result[3][2];
    printf("Enter the first matrix\n");
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 4; j++)
        {
            // printf("Enter the %d %d element of first matrix\n", i, j);
            scanf("%d", &a[i][j]);
            // printf("\t");
        }
        // printf("\n");
    }
    printf("Enter the second matrix\n");
    for (i = 0; i < 4; i++)
    {
        for (j = 0; j < 2; j++)
        {
            // printf("Enter the %d %d element of first matrix\n", i, j);
            scanf("%d", &b[i][j]);
            // printf("\t");
        }
        // printf("\n");
    }

    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 2; j++)
        {
            // Calculate the result
            for (k = 0; k < 4; k++)
            {
                sum += a[i][k] * b[k][j];
            }
            result[i][j] = sum;
            sum = 0;
        }
    }

    //Print the resultant matrix
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 2; j++)
```

```
        {
            // Print the result
            printf("%d \t", result[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

```
// Structure to create a node with data and the next pointer

struct Node {
    int data;
    struct Node *next;
};

Node* top = NULL;

int pop() {
    if (top == NULL) {
        printf("\nEMPTY STACK");
    } else {
        struct Node *temp = top;
        int temp_data = top->data; //to store data of top node
        top = top->next;
        free(temp); //deleting the node
        return temp_data;
    }
}
```

Write a C Program to implement Stack Operations Using Linkedlist.

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

// Structure to create a node with data and the next pointer
struct node {
   int info;
   struct node *ptr;
}*top,*top1,*temp;

int count = 0;
// Push() operation on a  stack
void push(int data) {
    if (top == NULL)
   {
      top =(struct node *)malloc(1*sizeof(struct node));
      top->ptr = NULL;
      top->info = data;
   }
   else
   {
      temp =(struct node *)malloc(1*sizeof(struct node));
      temp->ptr = top;
      temp->info = data;
      top = temp;
   }
   count++;
   printf("Node is Inserted\n\n");
}

int pop() {
    top1 = top;

   if (top1 == NULL)
   {
      printf("\nStack Underflow\n");
      return -1;
   }
   else
      top1 = top1->ptr;
   int popped = top->info;
```

```
    free(top);
    top = top1;
    count--;
    return popped;
}

void display() {
    // Display the elements of the stack
    top1 = top;

    if (top1 == NULL)
    {
        printf("\nStack Underflow\n");
        return;
    }

    printf("The stack is \n");
    while (top1 != NULL)
    {
        printf("%d--->", top1->info);
        top1 = top1->ptr;
    }
    printf("NULL\n\n");

}

int main() {
    int choice, value;
    printf("\nImplementation of Stack using Linked List\n");
    while (1) {
        printf("\n1. Push\n2. Pop\n3. Display\n4. Exit\n");
        printf("\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice) {
        case 1:
            printf("\nEnter the value to insert: ");
            scanf("%d", &value);
            push(value);
            break;
        case 2:
            printf("Popped element is :%d\n", pop());
            break;
        case 3:
            display();
```

```
            break;
        case 4:
            exit(0);
            break;
        default:
            printf("\nWrong Choice\n");
        }
    }
}
```