CYCLE 1

Cycle 1: Searching and Sorting

1. Write a program to search for an element in an array using Linear Search.

2. Write a program to sort a set of numbers using Selection sort and then search for a particular element in the array using binary search.

3. Write a program to sort a set of numbers using Insertion sort.

LINEAR SEARCH

```
#include<stdio.h>
void main(){
	int a[100],n,i,j,key;
	printf("LINER SEARCH\n");
	for(int k=0;k<12;k++){
		printf("%c",'-');
	}printf("\n");
	printf("Enter the size of the array = ");
	scanf("%d",&n);
	for(i=0;i<n;i++){
		printf("Enter the element at index position a%d = ",i);
		scanf("%d",&a[i]);
	}
	printf("Enter the number to be found = ");
	scanf("%d",&key);
	for(i=0;i<n;i++){
		if(key==a[i]){
			printf("%d is found at index position %d\n",key,i);
			break;
		}
	}
	if(i==n){
		printf("%d is not present in the array\n",key);
	}
}
```

SELECTION SORT

```
#include<stdio.h>
void main(){
	printf("SELECTION SORT WITH BINARY SEARCH\n");
	for(int k=0;k<35;k++){
		printf("%c",'-');
	}printf("\n");
	int a[100],n,i,j,small,swap,low,high,mid,key;
	printf("Enter the size of the array = ");
	scanf("%d",&n);
	printf("Enter the array elements\n");
	for(i=0;i<n;i++){
		printf("a%d = ",i);
```

```
        scanf("%d",&a[i]);
    }
    for(i=0;i<n;i++){
        small=i;
        for(j=i+1;j<n;j++){
            if(a[small]>a[j]){
                small=j;
            }
        }
        if(small != i){
            swap=a[i];
            a[i]=a[small];
            a[small]=swap;
        }
    }
    printf("sorted elements in order\n");
    for(i=0;i<n;i++){
        printf("%d\n",a[i]);
    }
    printf("Enter the number to be found = ");
    scanf("%d",&key);

    low=0;
    high=n-1;
    mid=(low+high)/2;
    while(low <= high){
        if(key > a[mid]){
            low = mid+1;
        }
        else if(key == a[mid]){
            printf("%d is found at index position %d\n",key,i);
            break;
        }
        else{
            high=mid-1;
        }
    }
    if(low > high){
        printf("%d not present in the array\n",key);
    }
}
```

INSERTION SORT

```
#include<stdio.h>
void main(){
    printf("INSERTION SORT\n");
    for(int k=0;k<14;k++){
        printf("%c",'-');
    }printf("\n");
    int a[30],n,key,i=0;
    printf("Enter the size of the array = ");
    scanf("%d",&n);
```

```
	for(int i=0; i<n; i++){
		printf("a%d = ", i);
		scanf("%d",&a[i]);
	}
	printf("The array\n");
	for(int i=0; i<n; i++){
		printf("a%d = %d\n",i,a[i]);
	}
	for(int j=1;j<=n-1;j++){
		key=a[j];
		i=j-1;
		while(i>=0 && a[i]>=key){
			a[i+1]=a[i];
			i=i-1;
		}
		a[i+1]=key;
	}
	printf("The sorted array is\n");
	for(int i=0; i<n; i++){
		printf("a%d = %d\n",i,a[i]);
	}
	printf("\n");
}
```

CYCLE 2

Cycle 2: Applications of Array Data Structure - Polynomial Addition and Sparse Matrix Operations

1. Write a C program to read two polynomials and store them in an array. Find the sum of the two polynomials and display the input polynomials and the resultant sum polynomial.

2. Write a C program that reads a matrix in normal form , converts the matrix to tuple form and display it. Also find the transpose of matrix represented in tuple form and display it.

3. Write a C program that reads two matrices in normal form , converts them to tuple form and display them. Also find the sum of matrices represented in tuple form and display it.

POLYNOMIAL ADDITION

```
#include<stdio.h>
#define max 100
typedef struct pol{
	int coef;
	int exp;
}pol;
pol A[max];
void main(){
	int sA=0,sB,sC,fA,fB;
int i,c,p,q;
printf("POLYNOMIAL ADDITION\n");
for(i=0;i<20;i++){
        printf("%c",'-');
}printf("\n\n");
printf("No of terms in pol 1 = ");
scanf("%d",&p);
printf("No of terms in pol 2 = ");
scanf("%d",&q);
printf("\n");
sA=0;
fA=p-1;
sB=p;
fB=p+q-1;
sC=p+q;
for(i=0;i<p;i++){
        printf("Coef of pol 1 at a%d = ",i);
        scanf("%d",&A[i].coef);
        printf("Exp of pol 1 at  a%d = ",i);
        scanf("%d",&A[i].exp);
        printf("\n");
}
for(i=p;i<p+q;i++){
        printf("Coef of pol 2 at a%d = ",i);
        scanf("%d",&A[i].coef);
        printf("Exp of pol 2 at  a%d = ",i);
        scanf("%d",&A[i].exp);
        printf("\n");
}

while(sA<=fA && sB<=fB){
        if(A[sA].exp > A[sB].exp){
                A[sC].exp = A[sA].exp;
                A[sC].coef = A[sA].coef;
                sA++;
                sC++;
        }
        else if(A[sA].exp < A[sB].exp){
                A[sC].exp = A[sB].exp;
                A[sC].coef = A[sB].coef;
                sB++;
                sC++;
        }
        else{
                c = A[sA].coef + A[sB].coef;
                if(c != 0){
                        A[sC].exp = A[sB].exp;
                        A[sC].coef = c;
                        sC++;
                }
                sA++;
                sB++;
        }
    }
    while(sA<=fA){
        A[sC].exp = A[sA].exp;
        A[sC].coef = A[sA].coef;
        sA++;
        sC++;
    }
    while(sB<=fB){
        A[sC].exp = A[sB].exp;
        A[sC].coef = A[sB].coef;
        sB++;
        sC++;
    }

    printf("first polynomial  = ");
    for(i=0;i<p;i++){
        printf("%d x ^%d + ",A[i].coef, A[i].exp);
    }
    printf("\n");
    printf("second polynomial = ");
    for(i=p;i<p+q;i++){
        printf("%d x ^%d + ",A[i].coef, A[i].exp);
    }
    printf("\n");
    printf("The values of index positions\n sA=%d\n fA=%d\n sB=%d\n fB=%d\n sC=
%d\n",sA,fB,sB,fB,sC);
    printf("added polynomial = ");
    for(i=p+q;i<sC;i++){
        printf("%d x^%d + ",A[i].coef, A[i].exp);
    }
    printf("\n");
}
```

SPARSE TRANSPOSE

```
#include<stdio.h>
#define max 100
typedef struct{
    int row;
    int col;
    int value;
}sparse;
sparse A[max];
sparse B[max];
void main(){
    int a[100][100],i,j,r,c,k=1,p=1;
    printf("SPARSE MATRIX\n");
    for(int i=0;i<15;i++){
        printf("%c",'-');
    }printf("\n");
    printf("Enter no of rows = ");
```

```
	scanf("%d",&r);
	printf("Enter no of cols = ");
	scanf("%d",&c);
	printf("\nEnter the array elements\n");
	for(i=0;i<r;i++){
		for(j=0;j<c;j++){
			scanf("%d",&a[i][j]);
		}
	}
	A[0].row = r;
	A[0].col = c;
	for(i=0;i<r;i++){
		for(j=0;j<c;j++){
			if(a[i][j] != 0){
				A[k].row = i;
				A[k].col = j;
				A[k].value = a[i][j];
				k++;
			}
		}
	}
	A[0].value = k-1;
	printf("\nThe sparse matrix\n");
	for(i=0;i<k;i++){
		printf("%d  %d  %d  \n",A[i].row, A[i].col, A[i].value);
	}
	B[0].row = A[0].col;
	B[0].col = A[0].row;
	B[0].value = A[0].value;
	for(i=0;i<=A[0].col;i++){
		for(j=1;j<=A[0].value;j++){
			if( A[j].col == i){
				B[p].col = A[j].row;
				B[p].row = A[j].col;
				B[p].value = A[j].value;
				p++;
			}
		}
	}
	printf("\nThe transpose form\n");
	for(i=0;i<=B[0].value;i++){
		printf("%d  %d  %d  \n",B[i].row,B[i].col,B[i].value);
	}
}
```

SPARSE MATRIX ADDITION

```
#include<stdlib.h>
#include <stdio.h>
#define max 100
typedef struct
```

```
{
	int row ;
	int col ;
	int value ;
}sparce;
sparce A[max],B[max],C[max];
void main()
{
	int i,j,r1,r2,c1,c2,x,n=1,m=1,sum=0,k=1,c=0;
	printf("Sparse matrix addition\n");
	for(int i=0;i<24;i++){
		printf("%c",'-');
	}printf("\n");
	printf("No of rows of 1st matrix = ");
	scanf("%d",&r1);
	printf("No of cols of 1st matrix = ");
	scanf("%d",&c1);
	A[0].row=r1;
	A[0].col=c1;
	printf("\nNo of rows of 2nd matrix = ");
	scanf("%d",&r2);
	printf("No of cols of 2nd matrix = ");
	scanf("%d",&c2);
	B[0].row=r2;
	B[0].col=c2;
	if(r1!=r2 || c1!=c2){
		printf("Matrix addition not poosible !!!\nexiting the program\n!");
		exit(0);
	}

	printf("\nEnter the elements of first matrix\n");
	for (i=0;i<r1;i++){
		for (j=0;j<c1;j++){
			scanf("%d",&x);
			if(x!=0){
				A[m].row=i;
				A[m].col=j;
				A[m].value=x;
				m++;
			}
		}
	}
	A[0].value=m-1;
	printf("Tuple form of 1st matrix\n");
	for(i=0;i<m;i++){
		printf("%d  %d  %d \n",A[i].row,A[i].col,A[i].value);
	}
	printf("\nEnter the elements of Second matrix\n");
	for (i=0;i<r2;i++){
		for (j=0;j<c2;j++){
			scanf("%d",&x);
			if(x!=0){
```

```
            B[n].row=i;
            B[n].col=j;
            B[n].value=x;
            n++;
        }
    }
}
B[0].value=n-1;
printf("Tuple form of 2nd matrix\n");
for(j=0;j<n;j++){
    printf("%d  %d  %d \n",B[j].row,B[j].col,B[j].value);
}
i=1;
j=1;
int p=A[0].value;
int q=B[0].value;
while(i<=p && j<=q){
    if(A[i].row<B[j].row || A[i].col<B[j].col){
        C[k].col=A[i].col;
        C[k].row=A[i].row;
        C[k].value=A[i].value;
        i++;
        k++;
        c++;
    }
    else if(A[i].row>B[j].row || A[i].col>B[j].col){
        C[k].col=B[j].col;
        C[k].row=B[j].row;
        C[k].value=B[j].value;
        j++;
        k++;
        c++;
    }
    else{
        sum=A[i].value + B[j].value;
        if(sum!=0){
            C[k].col=A[i].col;
            C[k].row=A[i].row;
            C[k].value=sum;
            i++;
            k++;
            c++;
            j++;
        }
        else{
            i++;
            j++;
        }
    }
}
while(i<=p){
    C[k].col=A[i].col;
```

```
        C[k].row=A[i].row;
        C[k].value=A[i].value;
        i++;
        k++;
        c++;
    }
    while(j<=q){
        C[k].col=B[j].col;
        C[k].row=B[j].row;
        C[k].value=B[j].value;
        j++;
        k++;
        c++;
    }
    C[0].value=c;
    C[0].row=r1;
    C[0].col=c1;
    printf("\nAdded matrix in tuple form\n");
    for(i=0;i<k;i++){
        printf("%d  %d  %d \n",C[i].row,C[i].col,C[i].value);
    }
}
```

CYCLE 3

Cycle 3: Stacks and Queues

1. Write a menu driven C program to implement stack data structure using arrays.

2. Write a C program to convert an infix expression to its postfix equivalent and hence evaluate it .

3. Implement a Queue using arrays with the operations:
a) Insert elements to the Queue.
b) Delete elements from the Queue.
c) Display the contents of the Queue after each operation.

4. Implement a Circular Queue using arrays with the operations:
a) Insert elements to the Queue.
b) Delete elements from the Queue.
c) Display the contents of the Queue after each operation.

STACK

```
#include<stdio.h>
#include<stdlib.h>
void main(){
    int stack[10];
    int top=-1;
    int item;
    int i,max;
    int op=1;
    printf("STACK OPERATIONS\n");
    for(i=0;i<18;i++){
        printf("%c",'-');
    }printf("\n");
    printf("Enter the size of stack = ");
    scanf("%d",&max);
    while(op<4){
        printf("\n1.push operation\n2.pop operation\n3.Print the current stack\n4.exit\n");
        printf("Choice = ");
        scanf("%d",&op);
        printf("\n");
        switch(op){
        case 1 : if(top == max-1){
                printf("stack overflow/stack is full\n");
                break;
            }else{
                printf("Enter the number to push = ");
                scanf("%d",&item);
                top=top+1;
                stack[top]=item;
            }break;
        case 2 : if(top == -1){
                printf("stack underflow/stack is empty\n");
                break;
            }else{
                item=stack[top];
                top=top-1;
                printf("The element that poped = %d\n",item);
                break;
            }break;
        case 3 : printf("The current stack\n");
            if(top == -1){
                printf("stack underflow/empty\n");
                break;
            }else{
                for(i=0;i<=top;i++){
                    printf("a%d = %d\n",i,stack[i]);
                }
                printf("\n");
            }break;
        case 4 : printf("exiting the program...\n");
            exit(0);
        default:printf("Something wrong!!!\nexiting the program\n");
            exit(0);
    }}
}
```

QUEUE

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

```
void main(){
    int i,max;
    int Q[100];
    int front = -1;
    int rear = -1;
    int item;
    int ch=1;
    printf("QUEUE OPERATIONS\n");
    for(i=0;i<17;i++){
        printf("%c",'-');
    }
    printf("\n");
    printf("Enter size of the Queue = ");
    scanf("%d",&max);
    while(ch<5){
        printf("1.Enqueue\n2.Dequeue\n3.Print the Queue\n4.exit\n\n");
        printf("Enter your choice = ");
        scanf("%d",&ch);
        switch(ch){
            case 1 : if(rear == max-1){
                        printf("Queue overflow/Queue ended\nredirecting to main
menu...\n\n");
                        break;
                    }
                    else if(front==-1 && rear==-1){
                        front=rear=0;
                        printf("Enter the number to insert first = ");
                        scanf("%d",&item);
                        Q[rear]=item;
                    }
                    else{
                        rear = rear+1;
                        printf("Enter the number to insert = ");
                        scanf("%d",&item);
                        Q[rear] = item;
                    }
                    break;
            case 2 : if(front==-1 && rear==-1){
                        printf("Queue underflow/Queue is empty\nredirecting to main
menu...\n\n");
                    }
                    else if(front==rear){
                        item = Q[front];
                        printf("The last element deleted = %d\n",Q[front]);
                        front=rear=-1;
                    }
                    else{
                        item = Q[front];
                        printf("The element that deleted = %d\n",Q[front]);
                        front = front+1;
                    }
                    break;
            case 3 : if(front == -1){
                        printf("Queue is empty\nredirecting to main menu...\n\n");
                        break;
                    }
                    else{
                        printf("The current queue\n");
                        for(i=front;i<=rear;i++){
                            printf("%d\n",Q[i]);
                        }
                    }
                    printf("\n");
                    break;
            case 4 : printf("exiting the program...\n");
                    exit(0);
            default: printf("Something went wrong !!!\nprogram terminated...\n");
                    exit(0);
        }
    }
}
```

CIRCULAR QUEUE

```
#include<stdio.h>
#include<stdlib.h>
void main(){
    int i,max;
    int op=1;
    int front=-1;
    int rear=-1;
    int item;
    int Q[max];
    printf("CIRCULAR QUEUE OPERATIONS\n");
    for(int j=0;j<25;j++){
        printf("%c",'-');
    }
    printf("\n");
    printf("Enter size of circular Queue = ");
    scanf("%d",&max);
    while(op<4){
        printf("1.Enqueue\n2.Dequeue\n3.Print CircularQ\n4.exit\n\n");
        printf("Choice = ");
        scanf("%d",&op);
        switch(op){
            case 1 : if((rear+1)%max == front){
                    printf("Queue overflow/full\n");
                    break;
                }else if(front==-1){
                    front=rear=0;
                    printf("Enter the number to insert = ");
                    scanf("%d",&item);
                    Q[rear]=item;
                }else{
                    rear = (rear+1)%max;
                    printf("Enter the element to insert = ");
                    scanf("%d",&item);
                    Q[rear]=item;
                }break;
            case 2 : if(front==-1){
                    printf("Queue underflow/empty\n");
                }else if(front==rear){
                    item=Q[front];
                    printf("The last element deleted = %d\n",Q[front]);
                    front=rear=-1;
                }else{
                    item=Q[front];
                    printf("The element deleted = %d\n",Q[front]);
                    front=(front+1)%max;
                }break;
            case 3 : if(front==-1){
                    printf("Queue is empty\nredirecting to main menu...\n");
                }else{
                    printf("The current elememts in the circular Queue\n");
                    if(front<=rear){
                        for(i=front;i<=rear;i++){
                            printf("%d\n",Q[i]);
                        }
                    }else{
                        for(i=front;i<max;i++){
                            printf("%d\n",Q[i]);
                        }
                        for(i=0;i<=rear;i++){
                            printf("%d\n",Q[i]);
                        }
                    }
                    printf("\n");
                }break;
            case 4 : printf("exiting the program...\n");
                exit(0);
            default: printf("Something went wrong !!!\nterminating the program...\n");
                exit(0);
        }
    }
    printf("\n");
}
```

CYCLE 4

Cycle 4 : Deques, Priority Queues

1. Implement a Priority Queue using arrays with the operations:

a) Insert elements to the Queue.

b) Delete elements from the Queue.

c) Display the contents of the Queue after each operation.

2. Implement a Double-Ended Queue (DEQUE) with the operations:

a) Insert elements to the Front of the queue.
b) Insert elements to the Rear of the queue
c) Delete elements from the Front of the queue.
d) Delete elements from the Rear of the queue.
e) Display the queue after each operation.

DOUBLE ENDED QUEUE

```
#include <stdio.h>
#include<stdlib.h>
#define max 5
int DQ[max];
int front =-1, rear = -1, item;
void insert_front(){
        if((front==0 && rear==max-1) || (front==rear+1)){
                printf("Overflow");
        }else if((front==-1) && (rear==-1)){
                front=rear=0;
                printf("Enter the element to insert = ");
                scanf("%d",&item);
                DQ[front]=item;
        }else if(front==0){
                front=max-1;
                printf("Enter the element to insert = ");
                scanf("%d",&item);
                DQ[front]=item;

        }else{
                front=front-1;
                printf("Enter the element to insert = ");
                scanf("%d",&item);
                DQ[front]=item;
        }
}
void insert_rear(){
        if((front==0 && rear==max-1) || (front==rear+1)){
                printf("Overflow");
        }else if((front==-1) && (rear==-1)){
                rear=0;
                printf("Enter the element to insert = ");
                scanf("%d",&item);
                DQ[rear]=item;
        }else if(rear==max-1){
                rear=0;
                DQ[rear]=item;
        }else{
                rear++;
                printf("Enter the element to insert = ");
                scanf("%d",&item);
                DQ[rear]=item;
        }
}
```

```
void display(){
        int i=front;
        printf("\ncurrent DEQ\n");
        while(i != rear){
                printf("%d ",DQ[i]);
                i=(i+1)%max;
        }
        printf("%d\n",DQ[rear]);
}

void delete_front(){
        if((front==-1) && (rear==-1)){
                printf("DEQ underflow");
        }else if(front==rear){
                printf("\nThe deleted element is %d", DQ[front]);
                front=-1;
                rear=-1;
        }else if(front==(max-1)){
                printf("\nThe deleted element is %d", DQ[front]);
                front=0;
        }else{
                printf("\nThe deleted element is %d", DQ[front]);
                front=front+1;
        }
}

void delete_rear(){
        if((front==-1) && (rear==-1)){
                printf("DEQ underflow");
        }else if(front==rear){
                printf("\nThe deleted element is %d", DQ[rear]);
                front=-1;
                rear=-1;
        }else if(rear==0){
                printf("\nThe deleted element is %d", DQ[rear]);
                rear=max-1;
        }else{
                printf("\nThe deleted element is %d", DQ[rear]);
                rear=rear-1;
        }
}

void main(){
        printf("DEQ operations\n");
        for(int k=0;k<15;k++){
                printf("%c",'-');
        }printf("\n");
        int op = 1;
        while(op<7){
                printf("\n1.insert front\n2.insert rear\n3.delete front\n4.delete
rear\n5.display\n6.exit\n\n");
                printf("Enter choice = ");
                scanf("%d",&op);
                switch(op){
                case 1 : insert_front();
                        break;
                case 2 : insert_rear();
                        break;
                case 3 : delete_front();
                        break;
                case 4 : delete_rear();
                        break;
                case 5 : display();
                        break;
                case 6 : printf("exiting the program...\n");
                        exit(0);
                default: printf("something went wrong...program terminated\n");
                        exit(0);
                }
        }
}
```

CYCLE 5

Cycle 5 : Linked list operations

1. Write a menu driven program for performing the following operations on a Linked List:

1.Display
2.Insert at Beginning
3.Insert at End
4.Insert at a specified Position
5.Delete from Beginning
6.Delete from End
7.Delete from a specified Position

2. Do the operations in Question No. 1 using a Doubly linked list

3. Do the operations in Question No. 1 using a circular linked list

SINGLE LINKED LIST

```
#include<stdio.h>
#include<stdlib.h>
struct node{
        int data;
        struct node *next;
};
struct node *head;
void insert_Front();
void insert_End();
void insert_Any();
void delete_Front();
void delete_End();
void delete_Any();
void print();
void main(){
    int choice = 1;
    while(choice<9){
        printf("SINGLE LINKED LIST OPERATIONS\n");
        for(int k=0;k<29;k++){
            printf("%c",'-');
        }printf("\n");
        printf("1.insert at beginning\n2.insert at end\n3.insert at any position\n4.delete from beginning\n5.delete from end\n6.delete from any position\n7.print\n8.exit\n\n");
        printf("Enter your choice = ");
        scanf("%d",&choice);
    switch(choice){
        case 1: insert_Front();
        break;
        case 2: insert_End();
        break;
        case 3: insert_Any();
        break;
        case 4: delete_Front();
        break;
        case 5: delete_End();
        break;
        case 6: delete_Any();
        break;
        case 7: print();
        break;
        case 8: exit(0);
        break;
        default:printf("invalid key...program terminated !!!\n");
        exit(0);
    }
}}
void insert_Front(){
    struct node *ptr;
    int item;
    ptr = (struct node *) malloc(sizeof(struct node *));
    if(ptr == NULL){
        printf("overflow\n");
    }else{
        printf("\nEnter value = ");
        scanf("%d",&item);
        ptr->data = item;
        ptr->next = head;
        head = ptr;
        printf("Node inserted successfully\n\n");
    }
}
void insert_End(){
    struct node *ptr,*temp;
    int item;
    ptr = (struct node*)malloc(sizeof(struct node));
```

```
        if(ptr == NULL){
                printf("overflow\n");
        }else{
                printf("Enter value = ");
                scanf("%d",&item);
                ptr->data = item;
                if(head == NULL){
                        ptr -> next = NULL;
                        head = ptr;
                        printf("Node inserted successfully\n\n");
                }else{
                        temp = head;
                        while (temp -> next != NULL){
                                temp = temp -> next;
                        }
                        temp->next = ptr;
                        ptr->next = NULL;
                        printf("Node inserted successfully\n\n");
                }
        }
}
void insert_Any(){
        int i,l,item;
        struct node *ptr, *temp;
        ptr = (struct node *) malloc (sizeof(struct node));
        if(ptr == NULL){
                printf("overflow\n");
        }else{
                printf("\nEnter element value = ");
                scanf("%d",&item);
                ptr->data = item;
                printf("Enter the location after which you want to insert = ");
                scanf("%d",&l);
                temp=head;
                for(i=0;i<l;i++){
                        temp = temp->next;
                        if(temp == NULL){
                                printf("insertion failed\n\n");
                                return;
                        }
                }
                ptr ->next = temp ->next;
                temp ->next = ptr;
                printf("Node inserted successfully\n\n");
        }
}
void delete_Front(){
        struct node *ptr;
        if(head == NULL){
                printf("underflow\n\n");
        }else{
                ptr = head;
        head = ptr->next;
        free(ptr);
        printf("Node deleted from the begining\n\n");
    }
}
void delete_End(){
    struct node *ptr,*ptr1;
    if(head == NULL){
        printf("underflow\n\n");
    }
    else if(head -> next == NULL){
        head = NULL;
        free(head);
        printf("Only node of the list deleted\n\n");
    }else{
        ptr = head;
        while(ptr->next != NULL){
            ptr1 = ptr;
            ptr = ptr ->next;
        }
        ptr1->next = NULL;
        free(ptr);
        printf("Deleted Node from the last ...\n\n ");
    }
}
void delete_Any(){
    struct node *ptr,*ptr1;
    int l,i;
    printf("Enter the location to delete node = ");
    scanf("%d",&l);
    ptr=head;
    for(i=0;i<l;i++){
        ptr1 = ptr;
        ptr = ptr->next;
        if(ptr == NULL){
            printf("deletion failed\n\n");
            return;
        }
    }
    ptr1 ->next = ptr ->next;
    free(ptr);
    printf("Deleted node %d ",l+1);
}
void print(){
    struct node *ptr;
    ptr = head;
    if(ptr == NULL){
        printf("underflow\n");
    }else{
        printf("The current Linked List\n\n");
        while (ptr!=NULL){
            printf("%d\n",ptr->data);
```

```
                ptr = ptr -> next;
            }
            printf("\n");
        }
}
```