**VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES**

**VIVEKANANDA SCHOOL OF INFORMATION TECHNOLOGY**



**BACHELOR OF COMPUTER APPLICATION**

**Practical-V DS Lab**

**BCA-174**

**Guru Gobind Singh Indraprastha University   
Sector - 16C Dwarka, Delhi – 110078**



**SUBMITTED TO:                                   SUBMITTED BY:**

**INDEX HAD TO BE REMOVED BEFORE UPLOADING SINCE IT HAD SIGNATURES**

**SORRY :)**

**1. WAP to implement following operations on 1D array**

**(i) Insertion**

**Code:**

#include <stdio.h>

void print\_array(int n, int arr[])

{

for (int i=0; i<n; i++)

{

printf("%d, ", arr[i]);

}

printf("\n");

}

int main()

{

int i, n = 6, k=2, array0[10] = {1, 22, 33, 65, 22, 54};

printf("Array before insertion is: \n");

print\_array(n, array0);

printf("\nInserting 25 at index 2\n");

for (i=n-1; i>=k; i--)

{

array0[i+1]=array0[i];

}

array0[k]=25;

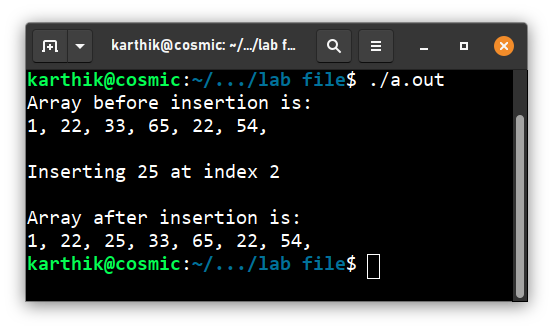
n=n+1;

printf("\nArray after insertion is: \n");

print\_array(n, array0);

}

**Output:**



(ii) Deletion

**Code:**

#include <stdio.h>

void print\_array(int n, int arr[])

{

for (int i=0; i<n; i++)

{

printf("%d, ", arr[i]);

}

printf("\n");

}

int main()

{

int i, n = 6, k=3, array0[10] = {1, 22, 33, 65, 22, 54};

printf("Array before deletion is: \n");

print\_array(n, array0);

printf("\nDeleting 65 at index 3\n");

for(i=k; i<n-1; i++)

{

array0[i] = array0[i+1];

}

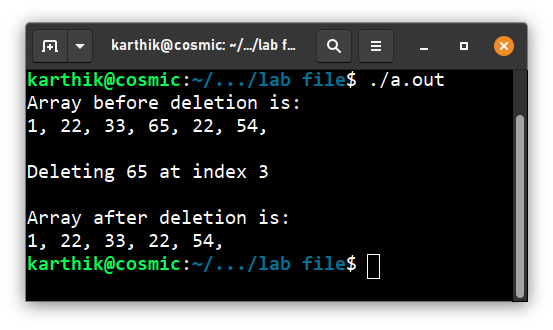
n=n-1;

printf("\nArray after deletion is: \n");

print\_array(n, array0);

}

**Output:**



(iii) Traversal

**Code :**

#include <stdio.h>

int main()

{

int i, n = 7, k=3, array0[10] = {1, 22, 33, 65, 29, 54, 92};

printf("\n Traversing through array0 and printing each element\n\n");

for(i=0; i<n; i++)

{

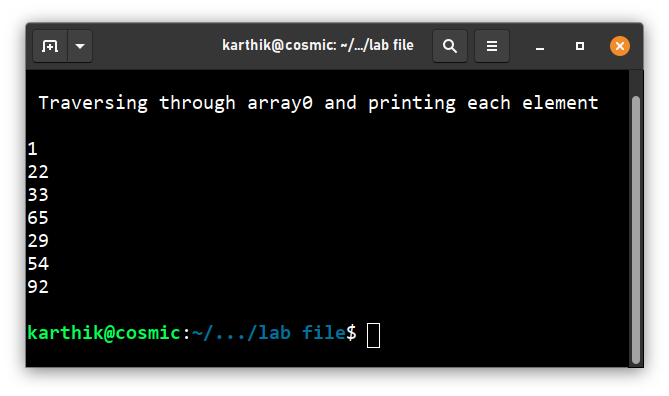
printf("%d\n", array0[i]);

}

printf("\n");

}

**Output :**



(iv) Reverse

**Code:**

#include <stdio.h>

void print\_array(int n, int arr[])

{

for (int i=0; i<n; i++)

{

printf("%d, ", arr[i]);

}

printf("\n");

}

int main()

{

int i, n = 6, array0[] = {1, 22, 33, 65, 22, 54};

printf("Array before reversing is: \n");

print\_array(n, array0);

for (i = 0; i < (n / 2); i++)

{

int temp = array0[i];

array0[i] = array0[(n - 1) - i];

array0[(n - 1) - i] = temp;

}

printf("\nArray after reversing is: \n");

print\_array(n, array0);

}

**Output :**



(v) Merge

**Code :**

#include <stdio.h>

void print\_array(int n, int arr[])

{

for (int i=0; i<n; i++)

{

printf("%d, ", arr[i]);

}

printf("\n");

}

int main(void)

{

int k=0, i=0, j=0, maxA=3, maxB=4, C[maxA+maxB+2];

int A[4] = {34, 44, 51, 62};

int B[5] = {11, 29, 57, 66, 71};

printf("Array A is : ");print\_array(maxA+1, A);

printf("\nArray B is : ");print\_array(maxB+1, B);

for (k=0; i<=maxA && j<=maxB;)

{

if (A[i]<=B[j])

{

C[k++]=A[i++];

}

else

{

C[k++]=B[j++];

}

}

while (i<=maxA)

{

C[k++]=A[i++];

}

while (j<=maxB)

{

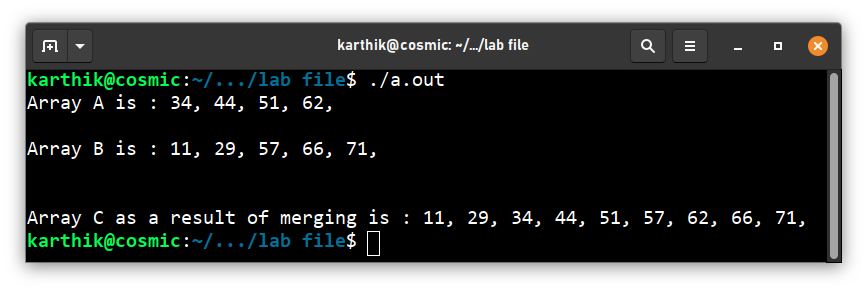
C[k++]=B[j++];

}

printf("\n\nArray C as a result of merging is : ");print\_array(maxA+maxB+2, C);

}

**Output :**



**2. WAP to sort an array using menu driven:**

**(i) Bubble Sort**

**(ii) Merge sort**

**(iii) Insertion Sort**

**(iv) Selection Sort**

**Code:**

#include <stdio.h>

void print\_array(int n, int arr[])

{

for (int i=0; i<n; i++)

{

printf("%d, ", arr[i]);

}

printf("\n");

}

void merge(int arr[], int p, int q, int r)

{

int b[9];

int i, j, k;

k = 0;

i = p;

j = q + 1;

while (i <= q && j <= r)

{

if (arr[i] <= arr[j])

{

b[k++] = arr[i++];

}

else

{

b[k++] = arr[j++];

}

}

while (i <= q)

{

b[k++] = arr[i++];

}

while (j <= r)

{

b[k++] = arr[j++];

}

for (i = r; i >= p; i--)

{

arr[i] = b[--k];

}

}

void mergesort(int arr[], int p, int r)

{

int q;

if (p < r)

{

q = (p + r) / 2;

mergesort(arr, p, q);

mergesort(arr, q + 1, r);

merge(arr, p, q, r);

}

}

int main()

{

int i, j, n=10, temp, min, pos;

int array0[10]={34,22,33,11,76,91,3,43,10,4};

printf("Array is : "); print\_array(10, array0);

int choice=0;

while (choice<5)

{

printf("\nEnter 1 for Bubble Sort\n");

printf("Enter 2 for Merge Sort\n");

printf("Enter 3 for Insertion sort\n");

printf("Enter 4 for selection Sort\n");

printf("Enter 5 to exit\n");

printf("Enter the choice : ");

scanf("%d", &choice);

switch (choice)

{

case 1:

{

int array0[10]={34,22,33,11,76,91,3,43,10,4};

printf("\nPerforming bubble sort\n");

n=10;

for (i=0; i<n; i++)

{

for (j=0; j<(n-1); j++)

{

if (array0[j]>array0[j+1])

{

temp=array0[j];

array0[j]=array0[j+1];

array0[j+1]=temp;

}

}

}

printf("Array after performing bubble sort is : ");

print\_array(10, array0);

break;

}

case 2:

{

int array0[10]={34,22,33,11,76,91,3,43,10,4};

printf("\nPerforming merge sort\n");

mergesort(array0, 0, n-1);

printf("Array after performing merge sort is : ");

print\_array(n, array0);

break;

}

case 3:

{

int array0[10]={34,22,33,11,76,91,3,43,10,4};

printf("\nPerforming insertion sort\n");

for (i=1; i<n; i++)

{

temp=array0[i];

j=i-1;

while ((j>=0) && (array0[j]>temp))

{

array0[j+1]=array0[j];

j--;

}

array0[j+1]=temp;

}

printf("Array after performing insertion sort is : ");

print\_array(n, array0);

break;

}

case 4:

{

int array0[10]={34,22,33,11,76,91,3,43,10,4};

printf("\nPerforming selection sort\n");

for (i=0; i<(n-1); i++)

{

min=array0[i];

pos=i;

for (j=i+1; j<=(n-1); j++)

{

if (array0[j]<min)

{

min=array0[j];

pos=j;

}

}

temp=array0[i];

array0[i]=array0[pos];

array0[pos]=temp;

}

printf("Array after performing selection sort is : ");

print\_array(n, array0);break;

}

case 5:

{

printf("Exiting...\n");break;

}

default:

{

printf("Invalid Input\n");

main();

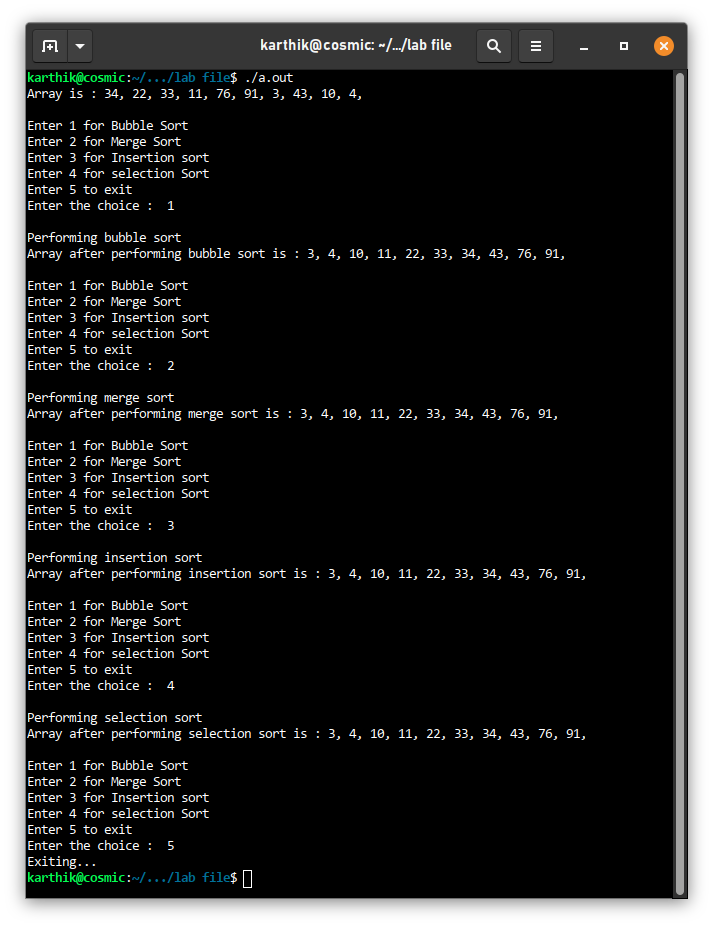
}

}

}

}

**Output:**



**3. WAP to search and display the location of an element specified by the user in an array**

**(i) Linear search**

**(ii) Binary search technique.**

**Code :**

#include <stdio.h>

void print\_array(int n, int arr[])

{

for (int i=0; i<n; i++)

{

printf("%d, ", arr[i]);

}

printf("\n");

}

int main()

{

int i, j, n=10, item\_to\_search;

int array0[10]= {3, 4, 10, 11, 22, 33, 34, 43, 76, 91};

printf("Array is : "); print\_array(10, array0);

int choice=0;

while (choice<3)

{

printf("\nEnter 1 for Linear Search\n");

printf("Enter 2 for Binary Search\n");

printf("Enter 3 to exit\n");

printf("Enter the choice : ");

scanf("%d", &choice);

switch (choice)

{

case 1:

{

printf("\nPerforming Linear Search\n");

printf("Enter element to be searched for : ");

scanf("%d", &item\_to\_search);

for (i=0; i<n; i++)

{

if (array0[i]==item\_to\_search)

{

printf("\nElement %d found at index %d\n", item\_to\_search, i);break;

}

}

if (i==n)

{

printf("\nElement %d not found!\n", item\_to\_search);

}

break;

}

case 2:

{

int start=0, end=n, mid;

printf("\nPerforming Binary Search\n");

printf("Enter element to be searched for : ");

scanf("%d", &item\_to\_search);

while (start<=end)

{

mid=(start+end)/2;

if (array0[mid]==item\_to\_search)

{

printf("\nElement %d found at index %d!\n", item\_to\_search, mid);

break;

}

else if (array0[mid]<item\_to\_search)

{

start=mid+1;

}

else

{

end=mid-1;

}

}

if (start>end)

{

printf("\nElement %d not found!\n", item\_to\_search);

}

break;

}

case 3:

{

printf("Exiting...\n");break;

}

default:

{

printf("Invalid Input\n");

main();

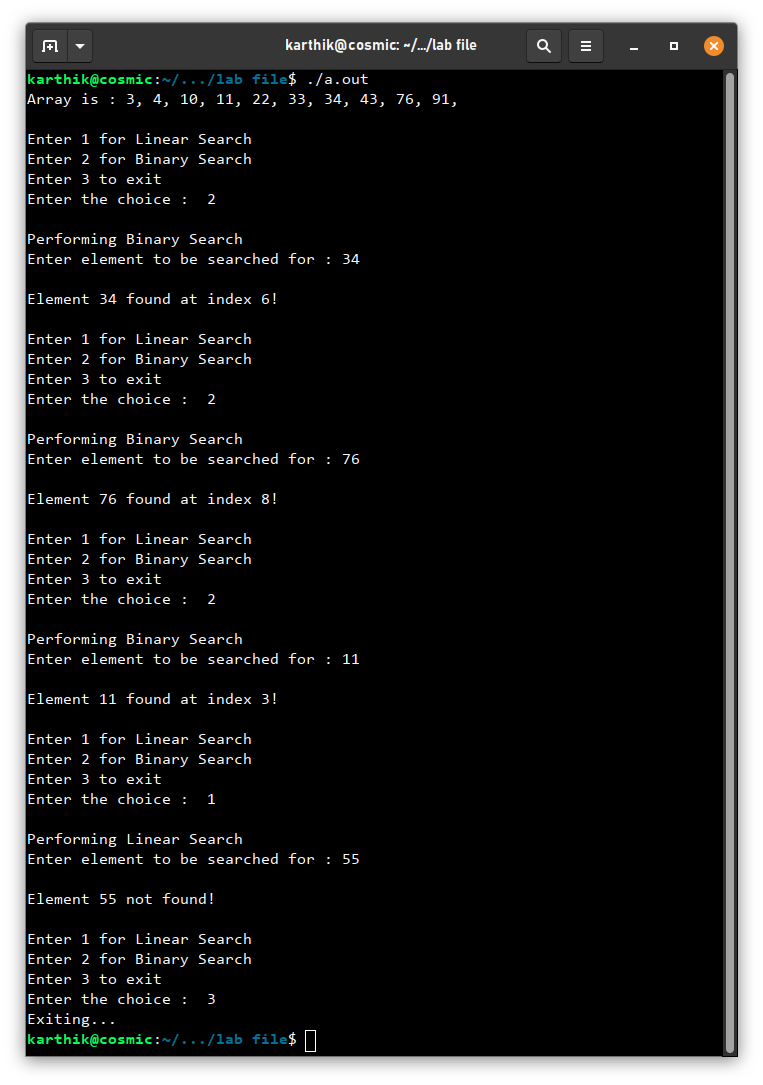
}

}

}

}

**Output :**



**4. WAP to accept a matrix from user, find out matrix is sparse or not and convert into triplex matrix.**

**Code:**

#include<stdio.h>

int main()

{

int a[10][10],b[20][3];

int m,n,i,j,count=0,k=1;

printf("Inputting matrix\n");

printf("\nEnter the number of rows of matrix:");

scanf("%d",&m);

printf("\nEnter the number of columns of matrix:");

scanf("%d",&n);

printf("\n");

for(i=0;i<m;i++)

{

for(j=0;j<n;j++)

{

printf("Enter the element of row-%d and column-%d: ",i+1,j+1);

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

}

}

printf("\n");

for(i=0;i<m;i++)

{

printf("| ");

for(j=0;j<n;j++)

{

printf("%d ",a[i][j]);

if(a[i][j]!=0)

{

count+=1;

}

}

printf(" |\n");

}

if(count<((m\*n)/2))

{

printf("\nThis matrix is sparse matrix.");

printf("\nnon-zero elements: %d",count);

printf("\nzero elements: %d",(m\*n)-count);

//Representing into 3 tuple form

b[0][0]=m;

b[0][1]=n;

for(i=0;i<m;i++)

{

for(j=0;j<n;j++)

{

if(a[i][j]!=0)

{

b[k][0]=i;

b[k][1]=j;

b[k][2]=a[i][j];

k++;

}

}

}

k--;

b[0][2]=k;

printf("\nThree tuple form\n");

for(i=0;i<=k;i++)

{

printf("| ");

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

{

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

}

printf(" |\n");

}

}

else

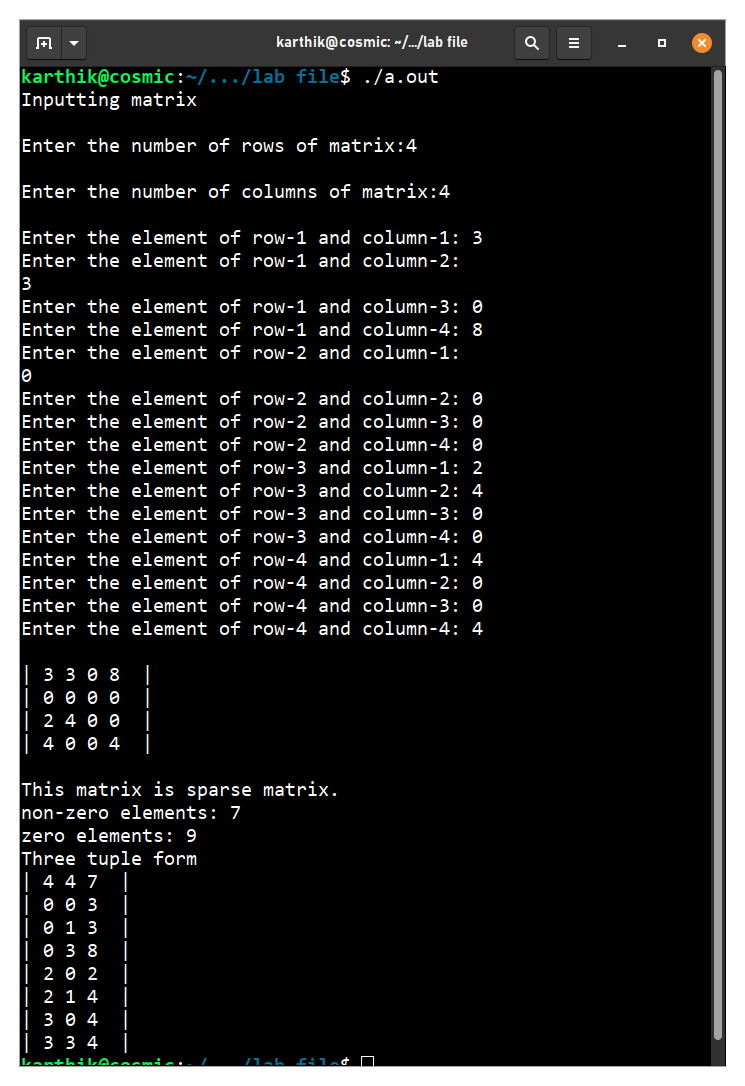
{

printf("\nThis matrix is not sparse matrix.");

}

}

**Output:**



**5. WAP to implement a Singly Linked List.**

**Code:**

#include<stdio.h>

#include<stdlib.h>

void create();

struct node{

int data;

struct node \*next;

};

struct node \*head,\*temp;

int main()

{

struct node \*newnode;

head=NULL;

int n,i;

printf("Enter no of nodes you want in a linked list: ");

scanf("%d",&n);

for(i=0;i<n;i++)

{

newnode=(struct node\*)malloc(sizeof(struct node));

printf("Enter value %d: ", i+1);

scanf("%d",&newnode->data);

newnode->next=NULL;

if(head==NULL)

{

temp=head=newnode;

}

else

{

temp->next=newnode;

temp=temp->next;

}

}

temp=head;

printf("Your list is : \n");

while(temp!=NULL)

{

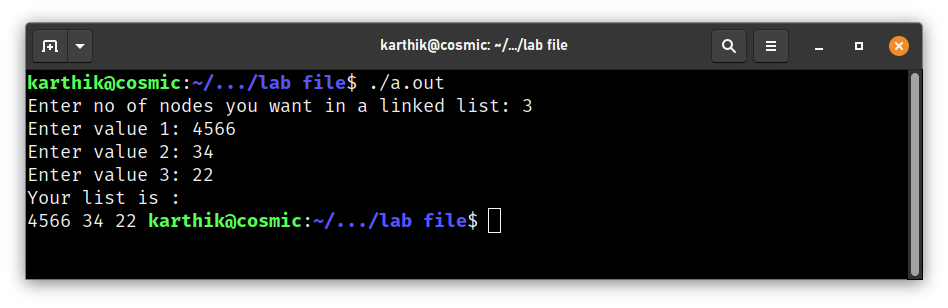
printf("%d ",temp->data);

temp=temp->next;

}

}

**Output:**

****

**6. WAP to implement Doubly Linked Lists.**

**Code:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next,\*prev;

};

struct node \*head,\*temp;

int main()

{

struct node \*newnode;

head=NULL;

int n,i;

printf("Enter no of nodes: ");

scanf("%d",&n);

for(i=0;i<n;i++)

{

newnode=(struct node\*)malloc(sizeof(struct node));

printf("Enter value %d: ", i+1);

scanf("%d",&newnode->data);

newnode->next=NULL;

if(head==NULL)

{

temp=head=newnode;

newnode->prev=NULL;

}

else

{

temp->next=newnode;

newnode->prev=temp;

temp=temp->next;

}

}

temp=head;

printf("Your list is : \n");

while(temp!=NULL)

{

printf("%d ",temp->data);

temp=temp->next;

}

}

**Output:**

****

**7. WAP to implement Circular Linked Lists.**

**Code:**

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

};

struct node \*head,\*temp;

int main()

{

struct node \*newnode;

head=NULL;

int n,i;

printf("Enter no of nodes: ");

scanf("%d",&n);

for(i=0;i<n;i++)

{

newnode=(struct node\*)malloc(sizeof(struct node));

printf("Enter value %d: ", i+1);

scanf("%d",&newnode->data);

if(head==NULL)

{

temp=head=newnode;

newnode->next=head;

}

else

{

temp->next=newnode;

newnode->next=head;

temp=newnode;

}

};

temp=head;

printf("Your list is : \n");

while(temp->next!=head)

{

printf("%d ",temp->data);

temp=temp->next;

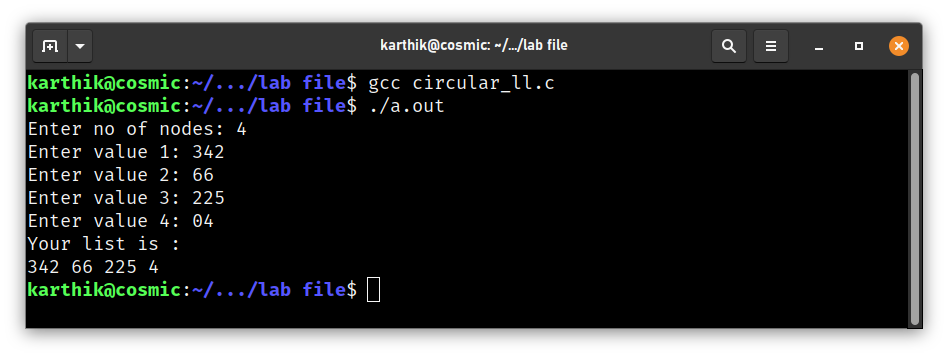
}

printf("%d ",temp->data);

printf("\n");

}

**Output:**



**8. WAP to implement a (i) Static Stack (ii) Dynamic Stack.**

**(i) Static Stack**

**Code:**

#include <stdio.h>

int main(){

int n,i,j;

printf("Enter the Size of stack : ");

scanf("%d",&n);

int arr[n];

int a = 1;

int k;

int item = -1;

while (a==1)

{

printf("STACK \n");

for(i = n-1;i>=0;i--){

printf("|\t%d\t|\n",arr[i]);

}

printf("1. Push\n");

printf("2. Pop\n");

printf("3. Exit\n");

printf("Enter Choice: ");

scanf("%d",&k);

if(k==1){

if(item >=n-1){

printf("Stack overflow\n");

break;

}

else{

printf("Enter the value at %d index stack : ",item+1);

scanf("%d",&arr[++item]);

}

}

else if(k==2){

if(item== -1){

printf("Stack underflow\n");

break;

}

else{

printf("Poped Element is %d ",arr[item]);

arr[item--] =0;

}

}

else if(k==3){

a=0;

}

else{

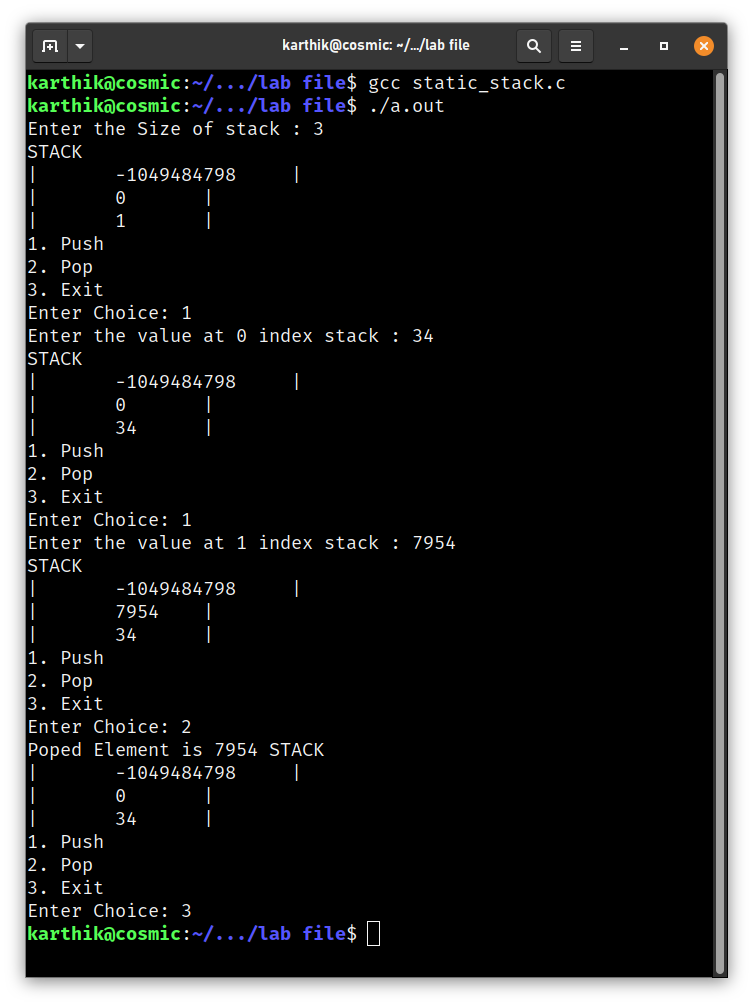
a=0;

}

}

}

**Output:**



**(ii) Dynamic Stack**

**Code:**

#include <stdio.h>

#include<stdlib.h>

void push(int x);

void pop();

void traverse();

struct node{

int data;

struct node \*link;

};

struct node \* top=NULL;

int main() {

int choice;

do{

printf("1. For push\n");

printf("2.For pop\n");

printf("3. For Traversal\n");

printf("4. For exit");

printf("Enter the choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

{

int item;

printf("enter item to push\n");

scanf("%d",&item);

push(item);

break;

}

case 2:

{

pop();

break;

}

case 3:

{

traverse();

break;

}

case 4:

{

choice=0;

break;

}

}

}while(choice!=0);

}

void push(int x)

{

struct node \*newnode;

newnode=(struct node \*)malloc(sizeof(struct node));

newnode->data =x;

newnode->link=top;

top=newnode;

}

void traverse()

{

struct node \*temp;

temp=top;

while(temp!=NULL)

{

printf("%d \n",temp->data);

temp=temp->link;

}

}

void pop()

{

struct node \*temp;

temp=top;

if (top==NULL)

printf("underflow");

else

{

printf("Popped element is %d ",top->data);

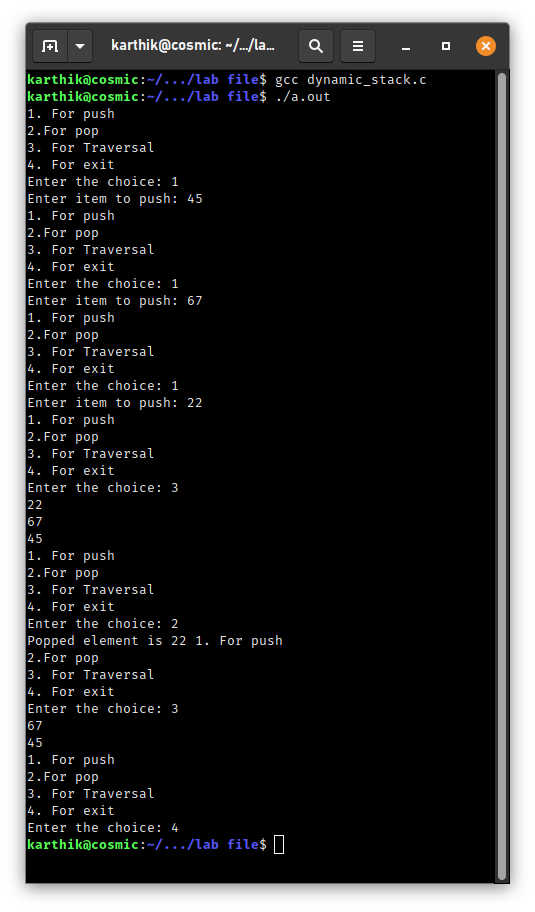
top=top->link;

free(temp);

}

}

**Output:**

****

**9. WAP to implement a (i) Static (ii) Dynamic Circular Queue.**

**(i) Static Circular Queue**

**Code:**

#include <stdio.h>

int main()

{

static int n = 4, queue[4], front=-1, rear=-1, item, i;

int choice;

printf("\nA Queue of size %d has been declared\n", n);

do

{

printf("\nEnter 1 to Enqueue\n");

printf("Enter 2 to Dequeue\n");

printf("Enter 3 to traverse\n");

printf("Enter 4 to exit\n");

printf("Enter choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

{

if ((front==rear+1)||((front==0)&&(rear==n-1)))

{

printf("Overflow\n");break;

}

else

{

if (front==-1)

{

front=rear=0;

}

else if(rear==n-1)

{

rear=0;

}

else ++rear;

printf("Enter element to be enqueued: ");

scanf("%d", &queue[rear]);

}break;

}

case 2:

{

if((front==-1)&&(rear==-1))

{

printf("Underflow\n");break;

}

else

{

item = queue[front];

if (front==rear)

{

front=rear=-1;

}

else if (front==n-1)

{

front=0;

}

else

{

front++;

}

printf("%d dequeued successfully\n", item);

}break;

}

case 3:

{

if((front==-1)&&(rear==-1))

{

printf("Queue is Empty. Nothing to print\n");break;

}

else if (front<=rear)

{

for (i = front; i <= rear; i++)

printf("%d, ", queue[i]);

}

else

{

for (i = front; i <= n-1; i++)

printf("%d, ", queue[i]);

for (i = 0; i <= rear; i++)

printf("%d, ", queue[i]);

}break;

}

case 4:

{

printf("\nExiting. . . \n");

break;

}

default:

printf("Invalid Input... Please Try Again\n");

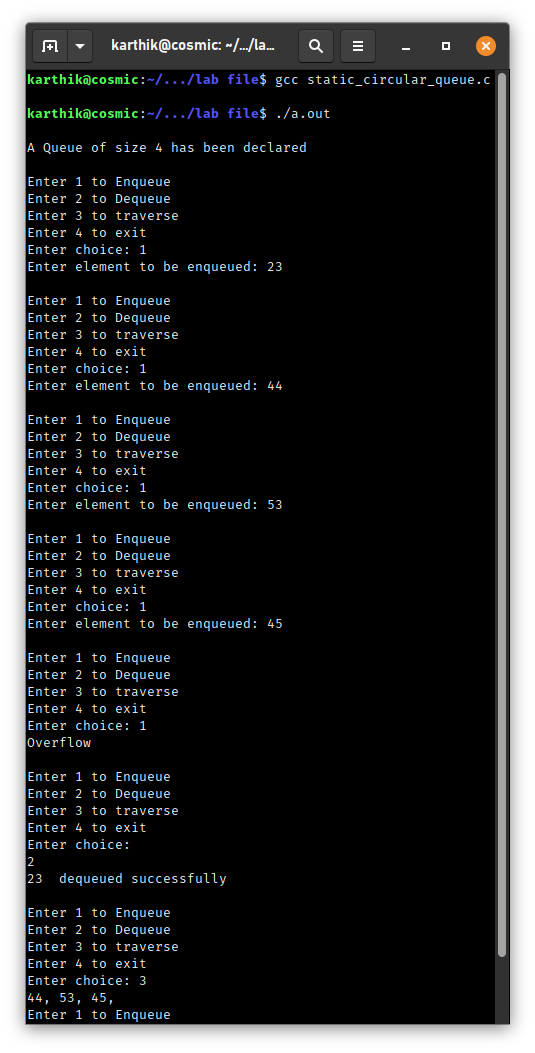
main();

}

}while(choice<4);

}

**Output:**



**(ii) Dynamic Circular Queue**

**Code:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\* next;

};

struct node \*f = NULL;

struct node \*r = NULL;

void enqueue(int d) //Insert elements in Queue

{

struct node\* n;

n = (struct node\*)malloc(sizeof(struct node));

n->data = d;

n->next = NULL;

if((r==NULL)&&(f==NULL))

{

f = r = n;

r->next = f;

}

else

{

r->next = n;

r = n;

n->next = f;

}

}

void dequeue() // Delete an element from Queue

{

struct node\* t;

t = f;

if((f==NULL)&&(r==NULL))

printf("\nQueue is Empty");

else if(f == r){

f = r = NULL;

free(t);

}

else{

f = f->next;

r->next = f;

free(t);

}

}

void print(){ // Print the elements of Queue

struct node\* t;

t = f;

if((f==NULL)&&(r==NULL))

printf("\nQueue is Empty");

else{

do{

printf("\n%d",t->data);

t = t->next;

}while(t != f);

}

}

int main()

{

int opt,n,i,data;

do{

printf("\n\n1 for Insert the Data in Queue\n2 for show the Data in Queue \n3 for Delete the data from the Queue\n0 for Exit");

printf("\nEnter Your Choice: ");

scanf("%d",&opt);

switch(opt){

case 1:

printf("\nEnter the number of data: ");

scanf("%d",&n);

i=0;

while(i<n){

printf("\nEnter value %d: ", i+1);

scanf("%d",&data);

enqueue(data);

i++;

}

break;

case 2:

print();

break;

case 3:

dequeue();

break;

case 0:

break;

default:

printf("\nIncorrect Choice");

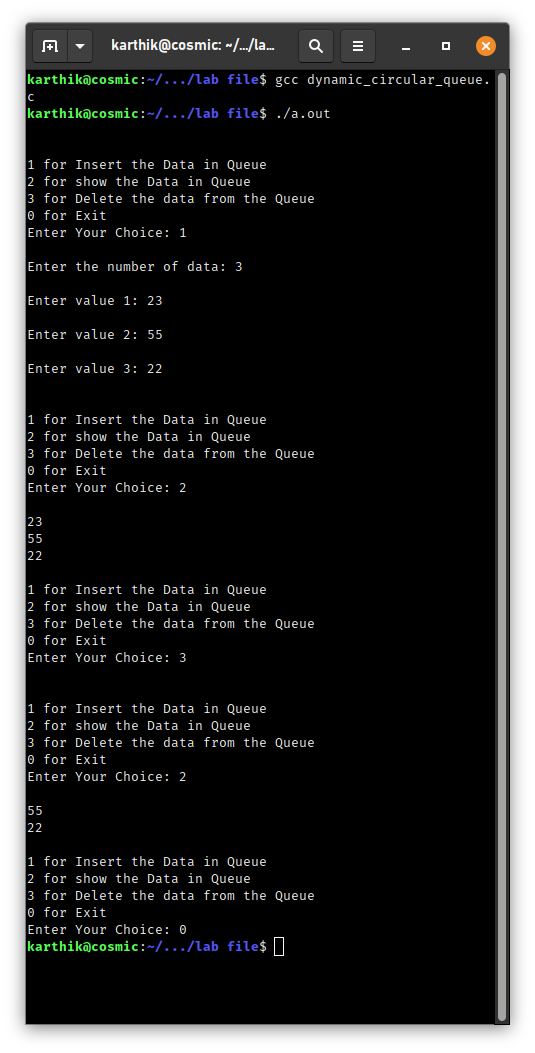
}

}while(opt!=0);

return 0;

}

**Output:**



**10. Write a program to convert infix to equivalent (i) Prefix Expression (ii) Postfix Expression**

**(i) Infix to Prefix**

**Code:**

#include<stdio.h>

#include<string.h>

#include<limits.h>

#include<stdlib.h>

# define MAX 100

int top = -1;

char stack[MAX];

// checking if stack is full

int isFull() {

return top == MAX - 1;

}

// checking is stack is empty

int isEmpty() {

return top == -1;

}

// Push function here, inserts value in stack and increments stack top by 1

void push(char item) {

if (isFull())

return;

top++;

stack[top] = item;

}

// Function to remove an item from stack. It decreases top by 1

int pop() {

if (isEmpty())

return INT\_MIN;

// decrements top and returns what has been popped

return stack[top--];

}

// Function to return the top from stack without removing it

int peek(){

if (isEmpty())

return INT\_MIN;

return stack[top];

}

// A utility function to check if the given character is operand

int checkIfOperand(char ch) {

return (ch>= 'a' &&ch<= 'z') || (ch>= 'A' &&ch<= 'Z');

}

// Fucntion to compare precedence

// If we return larger value means higher precedence

int precedence(char ch)

{

switch (ch)

{

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '^':

return 3;

}

return -1;

}

// The driver function for infix to postfix conversion

int getPostfix(char\* expression)

{

int i, j;

for (i = 0, j = -1; expression[i]; ++i)

{

// Here we are checking is the character we scanned is operand or not

// and this adding to to output.

if (checkIfOperand(expression[i]))

expression[++j] = expression[i];

// Here, if we scan character ‘(‘, we need push it to the stack.

else if (expression[i] == '(')

push(expression[i]);

// Here, if we scan character is an ‘)’, we need to pop and print from the stack

// do this until an ‘(‘ is encountered in the stack.

else if (expression[i] == ')')

{

while (!isEmpty(stack) && peek(stack) != '(')

expression[++j] = pop(stack);

if (!isEmpty(stack) && peek(stack) != '(')

return -1; // invalid expression

else

pop(stack);

}

else // if an opertor

{

while (!isEmpty(stack) && precedence(expression[i]) <= precedence(peek(stack)))

expression[++j] = pop(stack);

push(expression[i]);

}

}

// Once all inital expression characters are traversed

// adding all left elements from stack to exp

while (!isEmpty(stack))

expression[++j] = pop(stack);

expression[++j] = '\0';

}

void reverse(char \*exp){

int size = strlen(exp);

int j = size, i=0;

char temp[size];

temp[j--]='\0';

while(exp[i]!='\0')

{

temp[j] = exp[i];

j--;

i++;

}

strcpy(exp,temp);

}

void brackets(char\* exp){

int i = 0;

while(exp[i]!='\0')

{

if(exp[i]=='(')

exp[i]=')';

else if(exp[i]==')')

exp[i]='(';

i++;

}

}

void InfixtoPrefix(char \*exp){

int size = strlen(exp);

// reverse string

reverse(exp);

//change brackets

brackets(exp);

//get postfix

getPostfix(exp);

// reverse string again

reverse(exp);

}

int main()

{

printf("The infix is: ");

char expression[] = "((a/b)+c)-(d+(e\*f))";

printf("%s\n",expression);

InfixtoPrefix(expression);

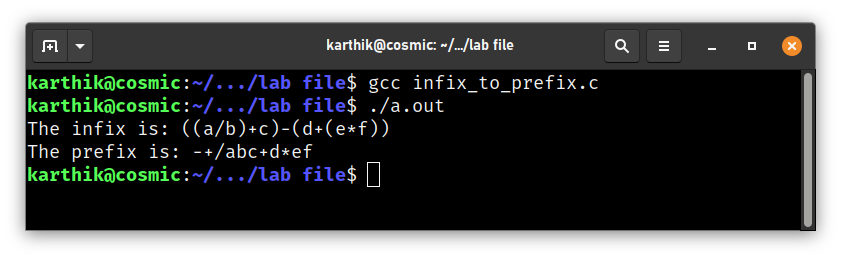
printf("The prefix is: ");

printf("%s\n",expression);

return 0;

}

**Output:**



**(ii) Infix to Postfix**

**Code:**

#include<stdio.h>

#include<ctype.h>

char stack[100];

int top = -1;

void push(char x)

{

stack[++top] = x;

}

char pop()

{

if(top == -1)

return -1;

else

return stack[top--];

}

int priority(char x)

{

if(x == '(')

return 0;

if(x == '+' || x == '-')

return 1;

if(x == '\*' || x == '/')

return 2;

return 0;

}

int main()

{

char exp[100];

char \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

e = exp;

while(\*e != '\0')

{

if(isalnum(\*e))

printf("%c ",\*e);

else if(\*e == '(')

push(\*e);

else if(\*e == ')')

{

while((x = pop()) != '(')

printf("%c ", x);

}

else

{

while(priority(stack[top]) >= priority(\*e))

printf("%c ",pop());

push(\*e);

}

e++;

}

while(top != -1)

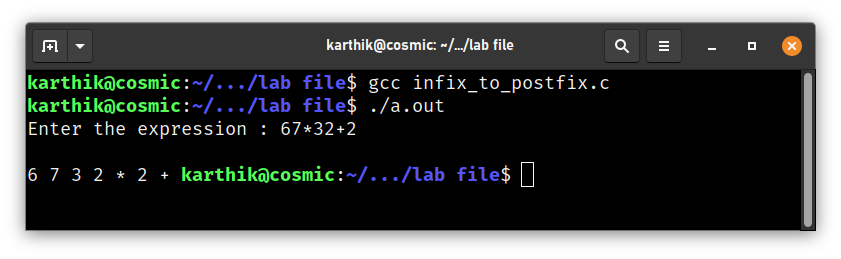
{

printf("%c ",pop());

}return 0;

}

**Output:**



**11. Write a program to evaluate (i) Prefix Expression (ii) Postfix Expression.**

**(i) Evaluate Prefix Expression**

**Code:**

#include<stdio.h>

#include<conio.h>

void main()

{

int s[30],i,a,b,top=-1;

char p[30];

printf("\n Enter prefix Expression:\t");

gets(p);

strcat(strrev(p),"(");

printf("%s",p);

for(i=0;p[i]!='(';i++)

{

if(p[i]=='+'||p[i]=='-'||p[i]=='\*'||p[i]=='/'||p[i]=='^')

{

a=s[top];

top--;

b=s[top];

printf("\n%d\t%d",a,b);

switch(p[i])

{

case '+':

s[top]=a+b;

printf("\t%d",s[top]);

break;

case '-':

s[top]=a-b;

printf("\t%d",s[top]);

break;

case '\*':

s[top]=a\*b;

printf("\t%d",s[top]);

break;

case '/':

s[top]=a/b;

printf("\t%d",s[top]);

break;

case '^':

s[top]=a^b;

printf("\t%d",s[top]);

break;

}

}

else

{

top++;

s[top]=p[i]-48;

}

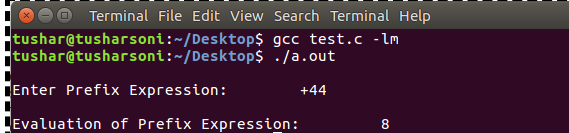
}

printf("Evaluation of Prefix expression:/t%d",s[top]);

getch();

}

**Output:**



**(ii) Evaluate Postfix Expression**

**Code:**

#include <stdio.h>

#include <math.h>

#include <string.h>

int main(){

char name[50];

int i,j;

int stack[50];

int top= -1;

int ref= 48;

int ex=0;

printf("Format : Ex - a,b,c,+,- here a, b , c can be two digit no 3 digit no or can be any digit number");

printf("Enter the Expression : ");

fgets(name,50,stdin);

for(i = 0;i<strlen(name)-1;){

int Fnum;

if(ex==1){

break;

}

else{

Fnum = name[i]-ref; // -2

for(j =1;j<strlen(name)-i-1;j++){ //25,

if(name[i+j]==',' || name[i+j] ==')'){

if(j==1){

// printf("%d",Fnum);

}

if(name[i+j] ==')'){

int ex = 1;

}

// printf("\n");

i = i+j+1;

break;

}

else{

int num = name[i+j] -ref;

Fnum = Fnum\*10 + num;

// printf("%d",Fnum);

}

}

// -4 = , ,\* = -6, - = -3 ,+ = -5,/ = -1 ,^ = 46,-7=),-8 =(

// printf("%d",Fnum);

if(Fnum==-1){// printf("/");

int num1 = stack[top--];

int num2 = stack[top];

stack[top] = num2 / num1;

}

else if(Fnum == -3){// printf("-");

int num1 = stack[top--];

int num2 = stack[top];

stack[top] = num2 - num1;

}

else if(Fnum== -5 ){// printf("+");

int num1 = stack[top--];

int num2 = stack[top];

stack[top] = num2 + num1;

}

else if(Fnum== -6){// printf("\*");

int num1 = stack[top--];

int num2 = stack[top];

stack[top] = num2 \* num1;

}

else if (Fnum == 46){// printf("^");

int num1 = stack[top--];

int num2 = stack[top];

stack[top] = pow(num2,num1);

}

else{

stack[++top] = Fnum;

}

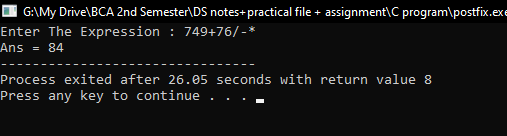
}

}

printf("Ans = %d",stack[0]);

}

**Output:**



**12. Write a C program to create two linked lists from a given list in the following way.**

**Input List :- 1 2 3 4 5 6 7 8 9 10**

**OUTPUT**

**First List:- 1 3 5 7 9**

**Second List:- 2 4 6 8 10**

**A-12**

**Code:**

#include <stdio.h>

#include <stdlib.h>

struct node {

int info;

struct node\* link;

};

struct node \*Head1,\*start1=NULL,\*newPointer,\*start2=NULL,\*start3 = NULL,\*Head2,\*Head3,\*p;

void Insertion(int n){

int i;

for( i=0;i<n;i++){

newPointer = (struct node \*) malloc(sizeof(struct node));

printf("Enter In the info part of the %d list",i+1);

scanf("%d",&newPointer->info);

newPointer->link = NULL;

if(start1 == NULL){

start1= Head1= newPointer;

}

else{

Head1->link= newPointer;

Head1 = Head1->link;

}

}

Head1 = start1;

}

void SeperatingOddAndEvenLinkedList(){

while(Head1 != NULL){

p = (struct node \*) malloc(sizeof(struct node ));

p->info = Head1->info;

p->link = NULL;

if(Head1->info % 2== 0){

if(start2 == NULL){

start2 = Head2 = p;

}

else{

Head2->link = p;

Head2 = Head2->link;

}

}

else if(Head1->info %2 != 0){

if(start3 == NULL){

start3 = Head3 = p;

}

else{

Head3->link = p;

Head3 = Head3->link;

}

}

Head1 = Head1->link;

}

Head2 = start2;

Head3 = start3;

}

void Display(struct node \*Head, struct node \* start){

printf("[Start | %p] ----> ",start);

while(Head->link!=NULL)

{

printf("[%d | %p] ----> ",Head->info,Head->link);

Head=Head->link;

}

printf("[%d | %p] \n\n",Head->info,Head->link);

}

int main(){

int n ;

printf("Enter The Number of Nodes :- ");

scanf("%d",&n);

Insertion(n);

Display(Head1,start1);

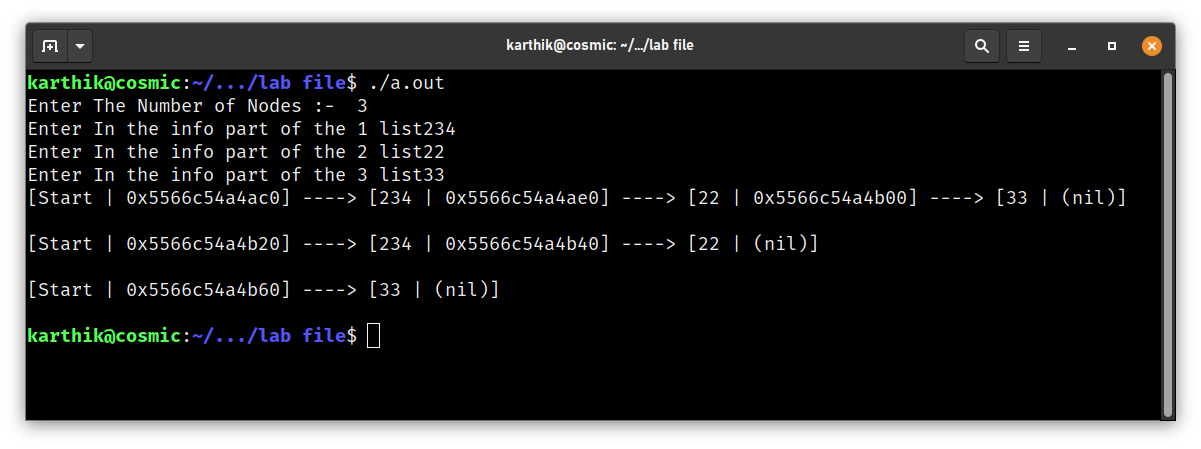
SeperatingOddAndEvenLinkedList();

Display(Head2,start2);

Display(Head3,start3);

}

**Output:**



**13. WAP to print type of sparse matrix.**

**Code:**

#include <stdio.h>

int main()

{

int m, i, j;

int value;

printf("Enter the Size of matrix :- ");

scanf("%d", &m);

int arr[m][m];

int b[m\*m];

int counterForB=0;

int k = 0,a=0;

while(a==0){

printf("1. Lower Triangular Sparse Matrix \n" );

printf("2. Upper Triangular Sparse Matrix \n" );

printf("3. Tri-diagonal Sparse Matrices \n" );

printf("4. Exit \n");

printf("Enter Here :- ");

scanf("%d",&value);

switch(value){

case 1:

for (i = 0; i< m; i++)

{

for (j = 0; j < m; j++)

{

if(i<j){

arr[i][j]=0;

k++;

}

else{

printf("Enter the value at Arr[%d,%d] : ", i,j);

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

b[counterForB++] = arr[i][j];

}

}

}

for (i = 0; i< m; i++)

{

printf(" \n| ");

for (j = 0; j < m; j++)

{

printf(" %d ", arr[i][j]);

}

printf(" | \n");

}

break;

case 2:

for (i = 0; i< m; i++)

{

for (j = 0; j < m; j++)

{

if(i>j){

arr[i][j]=0;

k++;

}

else{

printf("Enter the value at Arr[%d,%d] : ", i,j);

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

b[counterForB++] = arr[i][j];

}

}

}

for (i = 0; i< m; i++)

{

printf(" \n| ");

for (j = 0; j < m; j++)

{

printf(" %d ", arr[i][j]);

}

printf(" | \n");

}

break;

case 3:

for (i = 0; i< m; i++)

{

for (j = 0; j < m; j++)

{

if((i-j)<=1 && (i-j)>=-1){

printf("Enter the value at Arr[%d,%d] : ", i,j);

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

b[counterForB++] = arr[i][j];

}

else{

arr[i][j]=0;

k++;

}

}

}

for (i = 0; i< m; i++)

{

printf(" \n| ");

for (j = 0; j < m; j++)

{

printf(" %d ", arr[i][j]);

}

printf(" | \n");

}

break;

default:

a=1;

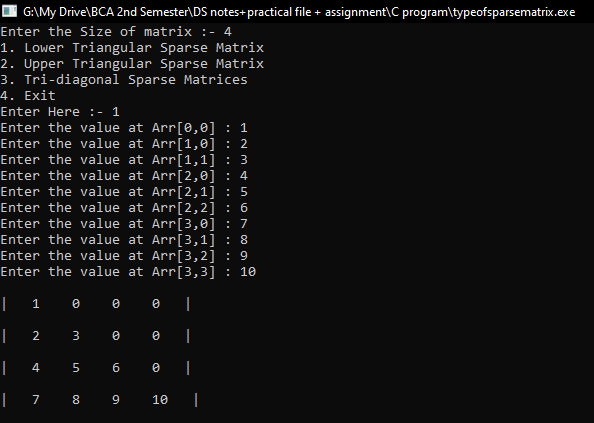
break;

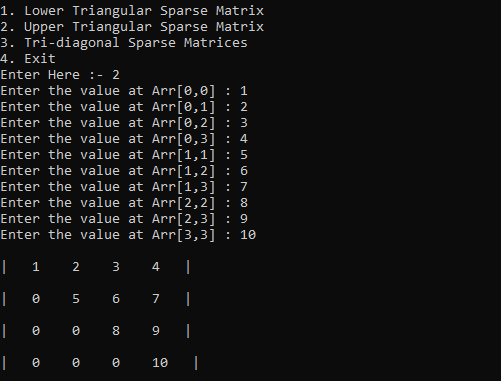
}

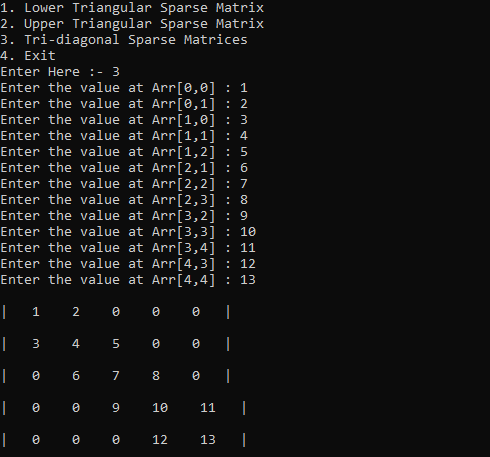
}

}

**Output:**







**14. WAP to represent different types of sparse matrix in 1 D array.**

**Code:**

#include <stdio.h>

int main()

{

int m, i, j;

int value;

printf("Enter the Size of matrix :- ");

scanf("%d", &m);

int arr[m][m];

int b[m\*m];

int counterForB=0;

int k = 0,a=0;

while(a==0){

printf("1. Lower Triangular Sparse Matrix \n" );

printf("2. Upper Triangular Sparse Matrix \n" );

printf("3. Tri-diagonal Sparse Matrices \n" );

printf("4. Exit \n");

printf("Enter Here :- ");

scanf("%d",&value);

switch(value){

case 1:

for (i = 0; i< m; i++)

{

for (j = 0; j < m; j++)

{

if(i<j){

arr[i][j]=0;

k++;

}

else{

printf("Enter the value at Arr[%d,%d] : ", i,j);

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

b[counterForB++] = arr[i][j];

}

}

}

for (i = 0; i< m; i++)

{

printf(" \n| ");

for (j = 0; j < m; j++)

{

printf(" %d ", arr[i][j]);

}

printf(" | \n");

}

printf("\nArray (in 1D):\n");

printf("\n[ ");

for(i=0;i<counterForB;i++)

{

printf(" %d ",b[i]);

}

printf(" ] \n\n");

break;

case 2:

for (i = 0; i< m; i++)

{

for (j = 0; j < m; j++)

{

if(i>j){

arr[i][j]=0;

k++;

}

else{

printf("Enter the value at Arr[%d,%d] : ", i,j);

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

b[counterForB++] = arr[i][j];

}

}

}

for (i = 0; i< m; i++)

{

printf(" \n| ");

for (j = 0; j < m; j++)

{

printf(" %d ", arr[i][j]);

}

printf(" | \n");

}

printf("\nArray (in 1D):\n");

printf("\n[ ");

for(i=0;i<counterForB;i++)

{

printf(" %d ",b[i]);

}

printf(" ] \n\n");

break;

case 3:

for (i = 0; i< m; i++)

{

for (j = 0; j < m; j++)

{

if((i-j)<=1 && (i-j)>=-1){

printf("Enter the value at Arr[%d,%d] : ", i,j);

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

b[counterForB++] = arr[i][j];

}

else{

arr[i][j]=0;

k++;

}

}

}

for (i = 0; i< m; i++)

{

printf(" \n| ");

for (j = 0; j < m; j++)

{

printf(" %d ", arr[i][j]);

}

printf(" | \n");

}

printf("\nArray (in 1D):\n");

printf("\n[ ");

for(i=0;i<counterForB;i++)

{

printf(" %d ",b[i]);

}

printf(" ] \n\n");

break;

default:

a=1;

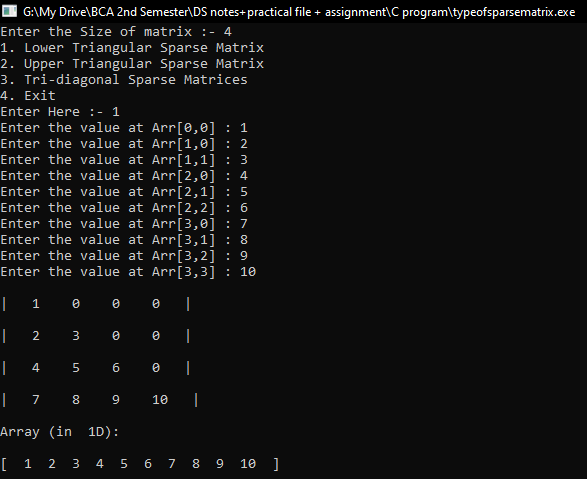
break;

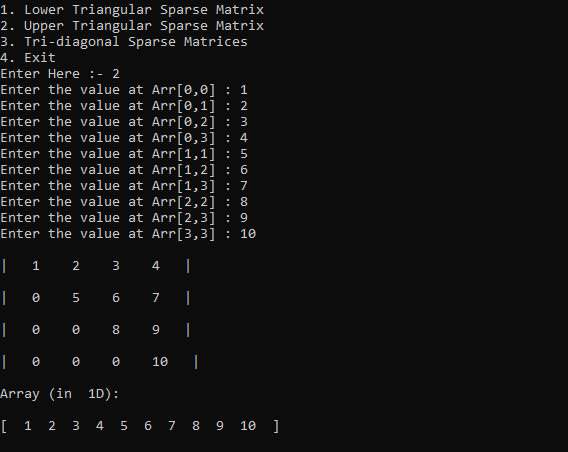
}

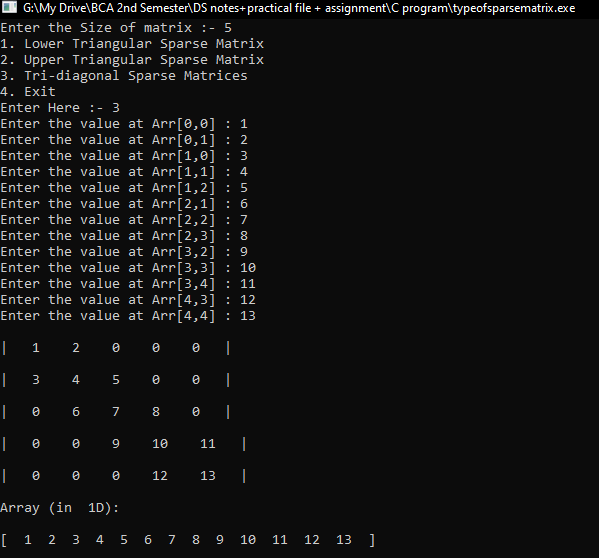
}

}

**Output:**







**15.Implement Static and Dynamic Dequeues**

**Static Dequeues:**

**Code:**

#include <stdio.h>

int main()

{

int size, i;

printf("Enter the size of deque");

scanf("%d", &size);

int deque[size];

int rare = -1;

int front = -1;

int stop = 0;

while (stop == 0)

{

printf("1.Insertion at Front of deque \n");

printf("2.Insertion at Rare of deque \n");

printf("3.Deletion from The start of deque \n");

printf("4.Deletion from The End of deque \n");

printf("5.Print the deque \n");

printf("6.Exit \n");

int ans;

printf("Enter here :- ");

scanf("%d", &ans);

int num;

switch (ans)

{

case 1:

if ((front == 0 && rare == size - 1) || (front == rare+ 1))

{

printf("Overflow");

}

else{

printf("Enter the Number To insert :- ");

scanf("%d",&num);

if ((front == -1) && (rare == -1))

{

front = rare = 0;

deque[front] = num;

}

else if (front == 0)

{

front = size - 1;

deque[front] = num;

}

else

{

deque[--front] = num;

}

}

printf("Inserted at front = %d rare = %d \n",front,rare);

printf("\n");

break;

case 2:

if((front ==0 && rare == size-1)||(rare+1== front)){

printf("OverFlow");

}

else{

printf("Enter the Number To insert :- ");

scanf("%d",&num);

if(front==-1 && rare ==-1){

front = rare = 0;

deque[rare] = num;

}

else if(rare==size-1){

rare = 0;

deque[rare] = num;

}

else{

deque[++rare]=num;

}

}

printf("Inserted at front = %d rare = %d \n",front,rare);

printf("\n");

break;

case 3:

if(rare == -1 && front == -1){

printf("UnderFlow");

}

else{

if(front ==rare){

printf("Deleted Element :- %d",deque[front]);

rare = front = -1;

}

else if(front == size-1){

printf("Deleted Element :- %d",deque[front]);

front = 0;

}

else{

printf("Deleted Element :- %d",deque[front++]);

}

}

printf("\n");

break;

case 4:

if(rare == -1 && front == -1){

printf("UnderFlow");

}

else{

if(front ==rare){

printf("Deleted Element :- %d",deque[rare]);

rare = front = -1;

}

else if(rare == 0){

printf("Deleted Element :- %d",deque[rare]);

rare = size-1;

}

else{

printf("Deleted Element :- %d",deque[rare--]);

}

}

printf("\n");

break;

case 5:

i = front;

while(i!=rare)

{

printf("%d ",deque[i]);

i=(i+1)%size;

}

printf("%d",deque[rare]);

printf("\n");

break;

default:

stop = 1;

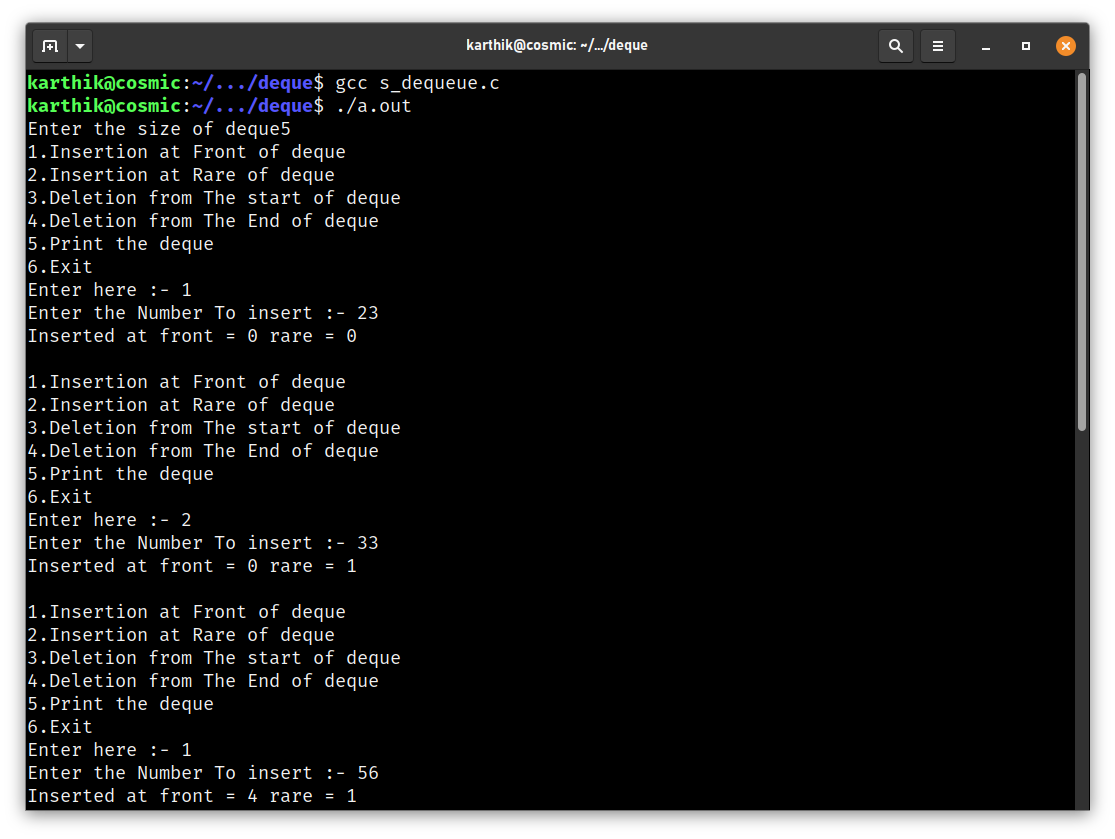
break;

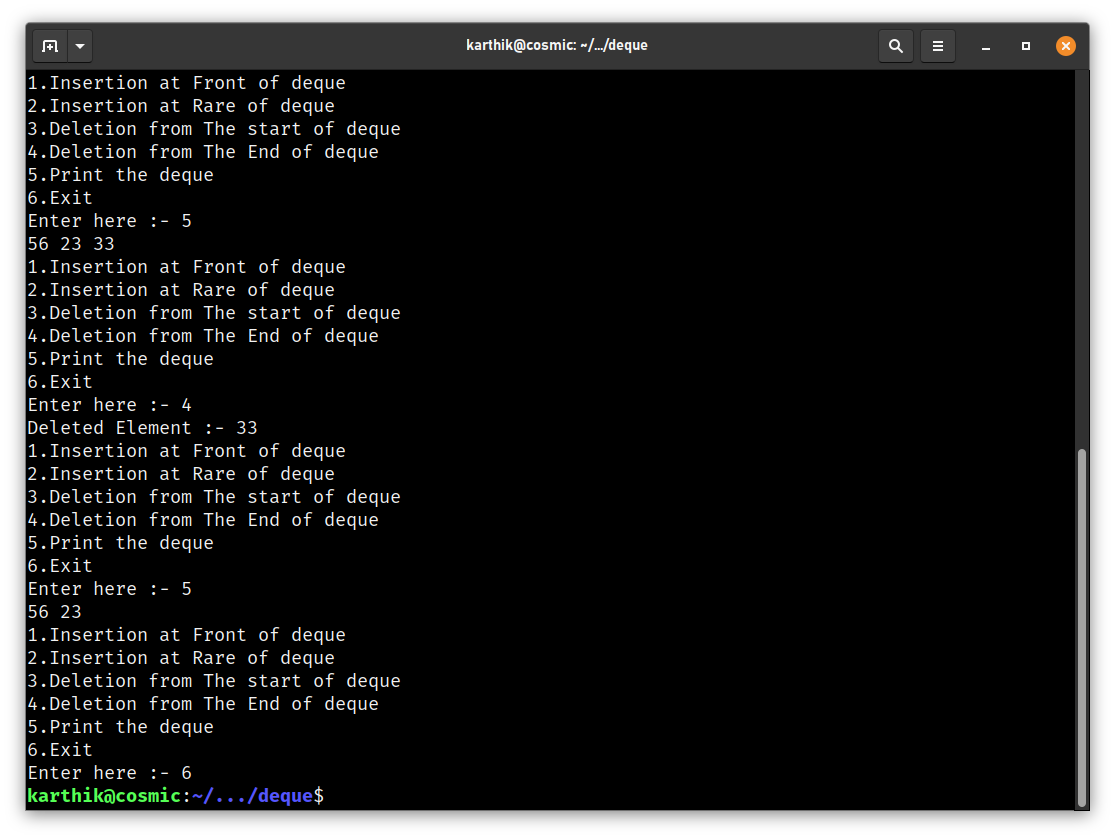
}

}

}

**Output:**





**Dynamic Dequeues:**

**Code:**

#include <stdio.h>

#include <stdlib.h>

struct node{ // struct of Node

int info;

struct node \* link;

};

struct node \* Rare,\*front,\*p,\*start,\* Head,\*temp;

void InsertionInTheFront(){

p = (struct node\*) malloc(sizeof(struct node)); // Address of the first node

printf("Enter in The Info Part :- "); // Adding info to that node

scanf("%d",&p->info);

if(start==NULL){ // If The node is first

start = p; // Start = p

Head = p; // Head = p

Head->link =NULL;

}

else{

p->link = Head;

Head =p;

}

}

void InsertionInTheRare(){

p = (struct node \*) malloc(sizeof(struct node));

printf("Enter the Info in the node");

scanf("%d",&p->info);

p->link=NULL;

if(start == NULL){

start = Head = front = Rare = p ;

}

else{

Head->link = p;

Rare =Head= p;

}

}

void DeletionFromTheRare(){

struct node \*previous;

while(Head!=NULL){

previous=Head;

Head = Head->link;

temp = Head;

}

previous->link = NULL;

free(temp);

}

void DeletionFromTheFront(){

Head = start;

temp = Head;

Head = Head->link;

start = Head;

printf("\nDeleted Node = [%d | %p]\n",temp->info,temp->link);

free(temp);

}

void Display(){

printf("List after insertion is : \n");

printf("[Start | %p] ----> ",start);

while(Head!=NULL)

{

printf("[%d | %p] ----> ",Head->info,Head->link);

Head=Head->link;

}

printf("[End]");

}

int main(){

int n=1,answer;

while(n==1){

printf("\n");

printf("1.Insertion From Front\n");

printf("2.Insertion From Rare\n");

printf("3.Deletion From Front\n");

printf("4.Deletion From Rare\n");

printf("5.Display\n");

printf("6.Exit\n");

printf("Enter the Answer:-");

scanf("%d",&answer);

switch (answer)

{

case 1:

InsertionInTheFront();

break;

case 2:

InsertionInTheRare();

break;

case 3:

DeletionFromTheFront();

break;

case 4:

DeletionFromTheRare();

break;

case 5:

Display();

break;

case 6:

n=0;

break;

default:

n=0;

break;

}

}

}

**Output:**

