**MASTER OF COMPUTER APPLICATIONS**

**2021-23**



**DATA AND FILE STRUCTURES**

**PRACTICAL FILE**

**Submitted to: Submitted by:**

**Mr. Rahul Mishra Nitish Gupta**

**M.C.A. (2nd Sem)**

**S.No: 28**

INDEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | LAB EXERCISES | TOPIC | DATE | SIGN |
| **1** | **LAB EXERCISE 1** | **MATRIX MULTIPLICATION** | **25-4-22** |  |
| **2** | **LAB EXERCISE 2** | **LINEAR SEARCH** | **25-4-22** |  |
| **3** | **LAB EXERCISE 3** | **BUBBLE SORT** | **25-4-22** |  |
| **4** | **LAB EXERCISE 4** | **MATRIX TRANSPOSE** | **25-4-22** |  |
| **5** | **LAB EXERCISE 5** | **STACK** | **2-5-22** |  |
| **6** | **LAB EXERCISE 6** | **QUEUE** | **2-5-22** |  |
| **7** | **LAB EXERCISE 7** | **FACTORIAL OF NUMBER** | **2-5-22** |  |
| **8** | **LAB EXERCISE 8** | **FIBONACCI SERIES** | **2-5-22** |  |
| **9** | **LAB EXERCISE 9** | **BINARY SEARCH** | **9-5-22** |  |
| **10** | **LAB EXERCISE 10** | **INSERTION SORT** | **9-5-22** |  |
| **11** | **LAB EXERCISE 11** | **SELECTION SORT** | **9-5-22** |  |
| **12** | **LAB EXERCISE 12** | **LINKED LIST** | **16-5-22** |  |
| **13** | **LAB EXERCISE 13** | **MERGE SORT** | **18-5-22** |  |
| **14** | **LAB EXERCISE 14** | **QUICK SORT** | **18-5-22** |  |
| **15** | **LAB EXERCISE 15** | **QUEUE USING STACK** | **23-5-22** |  |
| **16** | **LAB EXERCISE 16** | **BINARY TREE** | **25-5-22** |  |
| **17** | **LAB EXERCISE 17** | **BFS TRAVERSAL** | **13-6-22** |  |
| **18** | **LAB EXERCISE 18** | **DFS TRAVERSAL** | **13-6-22** |  |
| **19** | **LAB EXERCISE 19** | **PRIM’S ALGORITHM** | **20-6-22** |  |
| **20** | **LAB EXERCISE 20** | **KRUSKAL’S ALGORITHM** | **20-6-22** |  |
| **21** | **LAB EXERCISE 21** | **FILE HANDLING IN C** | **27-6-22** |  |
| **22** | **LAB EXERCISE 22** | **FILE HANDLING IN C** | **27-6-22** |  |
| **23** | **LAB EXERCISE 23** | **FILE HANDLING IN C** | **27-6-22** |  |
| **24** | **LAB EXERCISE 24** | **Floyd-Warshall Algorithm** | **4-7-22** |  |

**Q 1. Write a C Program to multiply two square matrices of the given order**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;

system("cls");

printf("enter the no. of rows=");

scanf("%d",&r);

printf("enter the no. ofolumn=");

scanf("%d",&c);

printf("enter the element of first matrix=\n");

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

{

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

{

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

}

}

printf("Enter the element of second matix=\n");

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

{

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

{

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

}

}

printf("Multipied Matrix\n");

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

{

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

{

mul[i][j]=0;

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

{

mul[i][j]+=a[i][k]\*b[k][j];

}

}

}

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

{

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

{

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

}

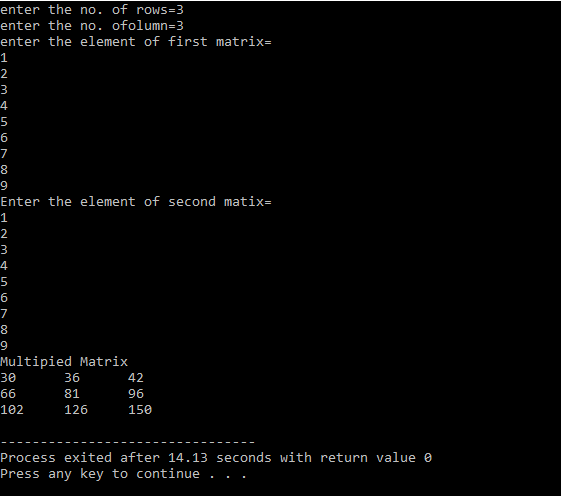
printf("\n");

}

return 0;

}

**OutPut:**



**Q 2. Write a C Program to implement Linear Search**

#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

int main()

{

int n;

printf("Enter the no. of elements in array=");

scanf("%d",&n);

int a[n],i,ele;

bool k=false;

printf("Enter the elements of array \n");

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

{

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

}

printf("Enter the element to be searched=");

scanf("%d",&ele);

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

{

if (a[i]==ele)

{

k=true;

}

}

if (k==true)

{

printf("Element Found");

}

else

{

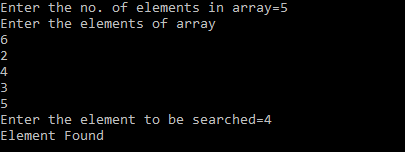
printf("Element not found");

}

return 0;

}

Output:



**Q 3. Write a C Program to implement Bubble Sort**

#include<stdio.h>

int main()

{

int n,i,j,temp;

printf("Enter the size of Array: \n");

scanf("%d", &n);

int a[n];

printf("Enter the elements of Array: \n");

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

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

printf("\n");

printf("\n");

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

{

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

{

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

{

temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

printf("Sorted array \n");

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

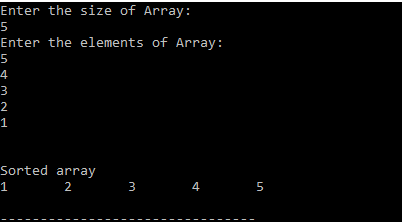
printf("%d\t", a[i]);

printf("\n");

return 0;

}

**Output:**

****

**Q 4. Write a C Program to transpose a square matrix of size nxn without using any auxiliary matrix**

#include<stdio.h>

int main()

{

int n,i,j,temp;

printf("Enter the order of Matrix: \n");

scanf("%d", &n);

int a[n][n];

printf("Enter the elements of matrix \n");

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

{

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

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

}

printf("\n");

printf("Original Matrix: \n");

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

{

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

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

printf("\n");

}

printf("\n");

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

{

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

{

temp = a[i][j];

a[i][j] = a[j][i];

a[j][i] = temp;

}

}

printf("Transposed Matrix: \n");

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

{

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

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

printf("\n");

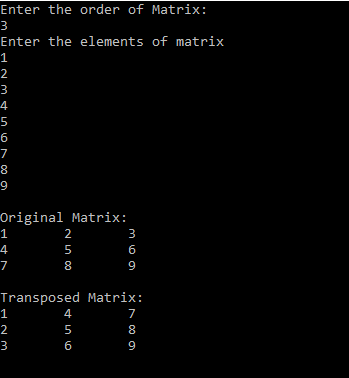
}

printf("\n");

return 0;

}

Output:



**Q 5. Write a C Program to implement a Stack and its operations using an array**

#include<stdio.h>

int top = -1, arr[100], i, element;

int push(int element,int n)

{

if(top == n-1)

printf("Stack Overflow");

else

{

arr[++top] = element;

printf("Element %d pushed successfully \n", element);

}

}

int pop()

{

int temp;

if(top == -1)

printf("Stack Underflow");

else

{

temp = arr[top];

top--;

printf("Element %d popped successfully \n", temp);

}

return temp;

}

int topElement()

{

if(top>=0)

return arr[top];

else

printf("No element is present in stack \n");

}

int empty()

{

if(top == -1)

printf("Stack is Empty \n");

else

printf("Stack is not empty \n");

}

void display()

{

if(top >=0)

{

printf("Elements of Stack are: \n");

for(i = top; i>=0;i--)

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

printf("\n");

}

}

int main()

{

int choice, n;

printf("Enter the size of Stack \n");

scanf("%d", &n);

do

{

printf("Stack Operations \n");

printf("1 - Push a Element \n");

printf("2 - Pop a Element \n");

printf("3 - For checking the emptyness of Stack \n");

printf("4 - For checking the top element of Stack \n");

printf("5 - Display the stack \n");

printf("6 - Exit \n");

printf("Thankyou For Using \n");

scanf("%d", &choice);

switch(choice)

{

case 1 :

printf("Enter the element to be pushed \n");

scanf("%d", &element);

push(element,n);

break;

case 2 :

pop();

break;

case 3 :

empty();

break;

case 4 :

printf("The element on top is %d \n", topElement());

break;

case 5 :

display();

break;

case 6 :

printf("No Operations to be performed now \n");

break;

default :

printf("Please enter a valid choice 1/2/3/4 \n");

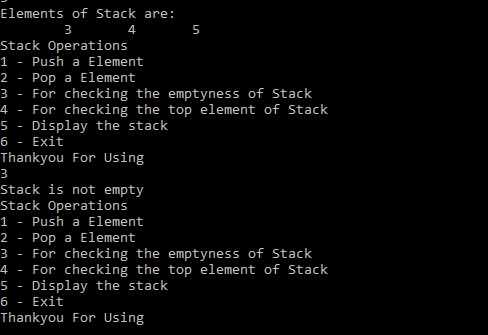
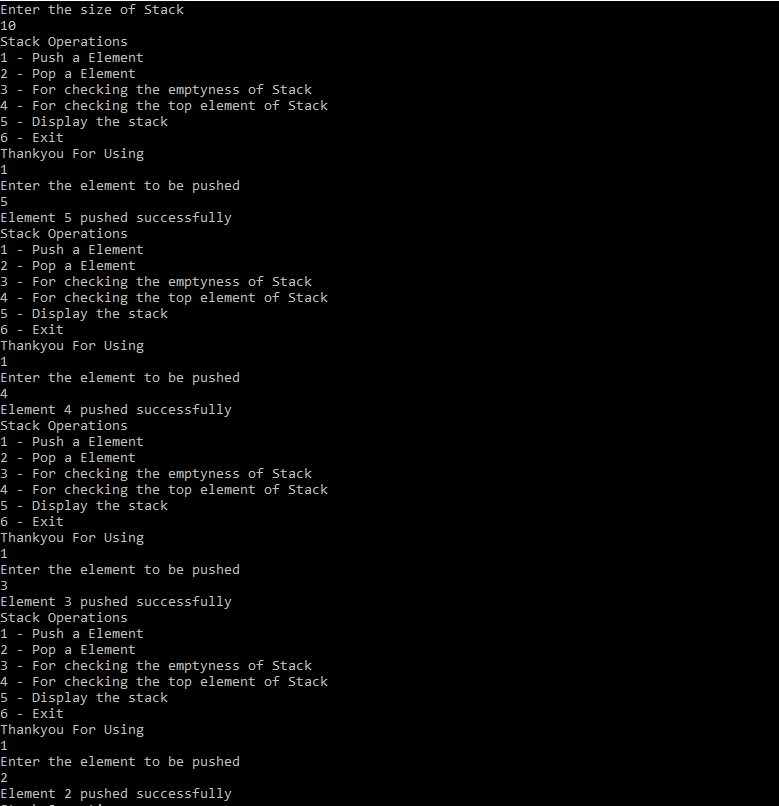
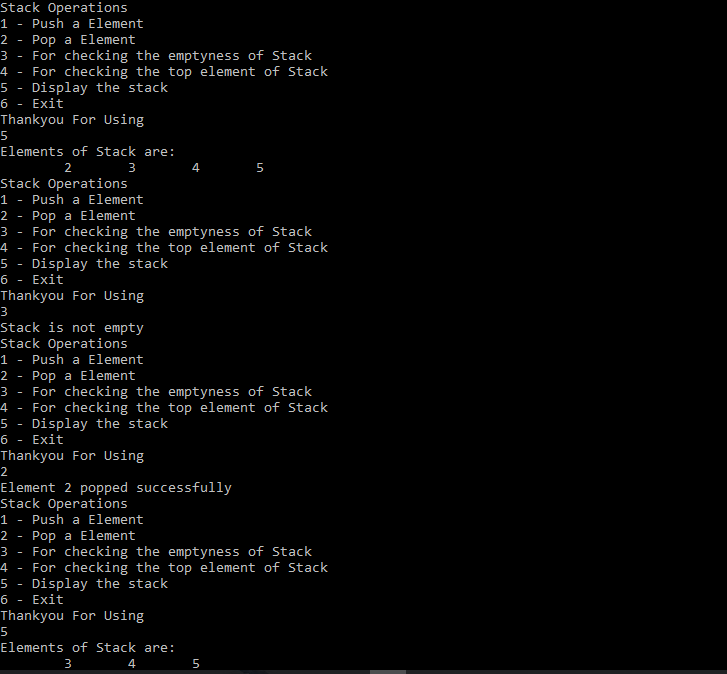
}

}while(choice!=6);

return 0;

}

Output:



**Q 6. Write a C Program to implement a Queue using an array**

#include<stdio.h>

#include<conio.h>

int arr[5],front=0,rear=0,n,capacity=5,elements=0;

void insert()

{

int x;

if(elements==capacity)

{

printf("queue overflow");

}

else

{

printf("Enter the item ");

scanf("%d",&x);

arr[rear]=x;

elements++;

rear=(rear+1)%10;

}

}

void Delete()

{

if(elements==0)

{

printf("queue underflow");

}

else

{

elements--;

front=(front+1)%10;

}

}

void display()

{

int i;

for(i=front;i!=rear;)

{

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

i=(i+1)%10;

}

printf("\n");

}

int main()

{

int ch=0;

while(ch==0)

{

printf("1.insert \n");

printf("2.Delete \n");

printf("3.Display \n");

printf("4.Exit \n");

scanf("%d",&n);

if(n==1)

insert();

else if(n==2)

Delete();

else if(n==3)

display();

else if (n==4)

{

ch=1;

printf("Thankyou for using \n");

}

else

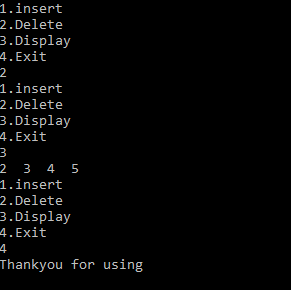
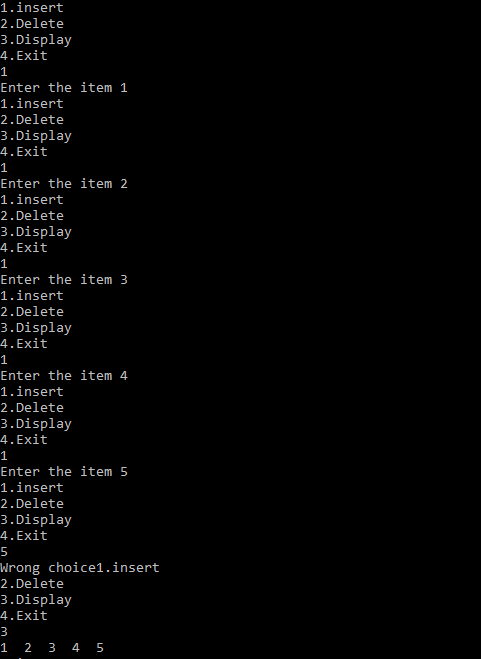
printf("Wrong choice");

}

return 0;

}

**Output:**

****

**Q 7. Write a C Program to find factorial of a given number n using:**

1. **Recursion**

#include<stdio.h>

int fact(int n)

{

if(n == 1)

return 1;

else

return n\*fact(n-1);

}

int main()

{

int n;

printf("Enter the number whose factorial is to be found \n");

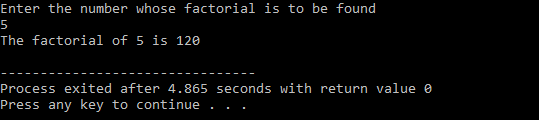
scanf("%d", &n);

printf("The factorial of %d is %d \n", n,fact(n));

return 0;

}

Output:



1. **Iteration**

#include<stdio.h>

int main()

{

int n,ans = 1,i;

printf("Enter the number whose factorial is to be found \n");

scanf("%d", &n);

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

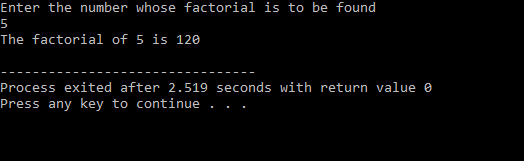
ans\*=i;

printf("The factorial of %d is %d \n", n,ans);

return 0;

}

Output:



**Q 8. Write a C Program to find Fibonacci term of a given index n using**

1. **Recursion**

#include<stdio.h>

int fib(int n)

{

if(n == 0 || n == 1)

return n ;

else

return fib(n-1) + fib(n-2);

}

int main()

{

int n;

printf("Enter the fibonacci term you want to find? \n");

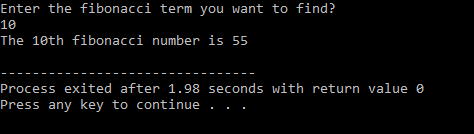
scanf("%d", &n);

printf("The %dth fibonacci number is %d \n", n, fib(n));

return 0;

}

**Output:**

****

**(ii) Iteration**

#include<stdio.h>

int main()

{

int n,i;

printf("Enter the fibonacci term you want to find? \n");

scanf("%d", &n);

int a = 0, b =1,c;

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

{

c= a + b;

a = b;

b = c;

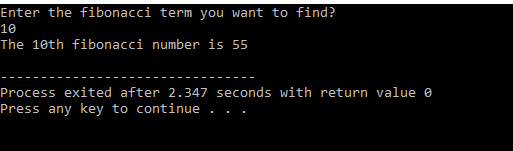
}

printf("The %dth fibonacci number is %d \n", n, c);

return 0;

}

**Output:**



**Q 9. Write a C Program to implement Binary Search**

//divide and conquer

#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

int main()

{

int n,low,high,mid;

printf("Enter the no. of elements in array=");

scanf("%d",&n);

int a[n],i,ele;

bool k=false;

printf("Enter the elements of array in sorted manner \n");

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

{

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

}

printf("Enter the element to be searched=");

scanf("%d",&ele);

low=0;

high=n-1;

int index=0;

while(low<=high)

{

mid=(low+high)/2;

if(ele==a[mid])

{

k=true;

index=mid;

}

if(ele>a[mid])

{

low=mid+1;

}

else

{

high=mid-1;

}

}

if (k==true)

{

printf("Element Found \n");

printf("Element found at the position %d\t",index+1);

}

else

{

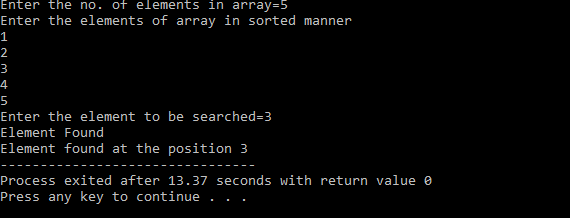
printf("Element not found");

}

return 0;

}

Output:



**Q 10. Write a C Program to implement Insertion Sort**

#include <stdio.h>

int main()

{

int n;

printf("Enter the size of array \n");

scanf("%d", &n);

int a[n],i,j,temp;

printf("Enter the elements of array \n");

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

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

// Insertion Sort

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

{

temp= a[i];

j = i - 1;

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

{

a[j + 1] = a[j];

j--;

}

a[j + 1] = temp;

}

printf("Sorted array \n");

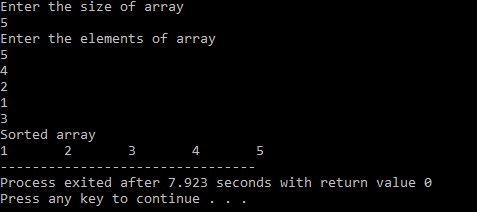
for (i = 0; i < n; i++)

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

return 0;

}

Output:



**Q 11. Write a C Program to implement Selection Sort**

#include<stdio.h>

int main()

{

int n,min,index,temp,i,j;

printf("Enter the size of array \n");

scanf("%d", &n);

int a[n];

printf("Enter the elements of array \n");

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

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

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

{

min = i;

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

{

if(a[j]<a[min])

{

min = j;

}

}

if(min!=i)

{

temp = a[i];

a[i] = a[min];

a[min] = temp;

}

}

printf("Sorted array \n");

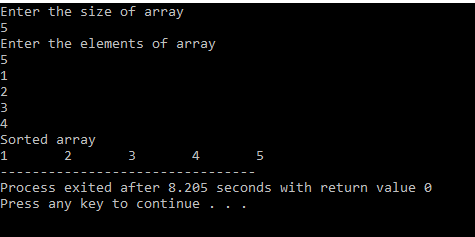
for(i = 0; i<n; i++)

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

return 0;

}

Output:



**Q 12. Write a C Program to**

**(i) To create a Linked List of Integers**

**(ii) To insert an integer at the end of the list**

**(iii) To delete first element of the list**

**(iv) To reverse the links of the given Linked List**

#include<stdio.h>

#include<stdlib.h>

// structure

struct node

{

int data;

struct node \*next;

};

// traversing

void display(struct node \*head)

{

struct node \*tmp=head;

if(tmp==NULL)

{

printf("No node found");

}

else

{

while(tmp!=NULL)

{

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

tmp=tmp->next;

}

}

printf("\n");

}

// Insertion at Beginning

struct node\* insertBeg(struct node \*head)

{

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

printf("Enter the data : ");

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

struct node \*tmp=head;

head=newnode;

newnode->next=tmp;

return head;

}

// Insertion at End

struct node\* insertEnd(struct node \*head)

{

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

printf("Enter the data : ");

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

struct node\* temp=head;

if(head==NULL)

{

head=newnode;

newnode->next=NULL;

}

else

{

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

newnode->next=NULL;

}

return head;

}

// Insert at Position

struct node\* insertPos(struct node \*head)

{

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

int pos,i=2;

printf("Enter the data : ");

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

printf("Enter the position : ");

scanf("%d",&pos);

struct node\* temp=head;

if(head==NULL)

{

head=newnode;

newnode->next=NULL;

}

else if(pos==1)

{

newnode->next=head;

head=newnode;

}

else

{

while(temp->next!=NULL&&i<pos)

{

temp=temp->next;

i++;

}

newnode->next=temp->next;

temp->next=newnode;

}

return head;

}

// Delete from Beginning

struct node\* deleteBeg(struct node\* head)

{

struct node\* temp=head;

if(head==NULL)

{

printf("No nodes present to delete !!!");

}

else

{

head=head->next;

free(temp);

}

display(head);

return head;

}

// Delete from End

struct node\* deleteEnd(struct node\* head)

{

struct node\* temp=head;

struct node\* ptr=head;

if(head==NULL)

{

printf("No nodes present to delete !!!");

}

else if(temp->next==NULL)

{

head=NULL;

free(temp);

}

else

{

while(temp->next!=NULL)

{

ptr=temp;

temp=temp->next;

}

ptr->next=NULL;

free(temp);

}

display(head);

return head;

}

// Delete from Position

struct node\* deletePos(struct node\* head)

{

struct node\* temp=head;

struct node\* ptr=head;

int pos,i=1;

printf("Enter the Position : ");

scanf("%d",&pos);

if(head==NULL)

{

printf("No nodes present to delete !!!\n");

}

else if(pos==1)

{

head=head->next;

free(temp);

}

else

{

while(temp->next!=NULL&&i<pos)

{

ptr=temp;

temp=temp->next;

i++;

}

if(i<pos)

{

printf("No node at that position !!\n");

}

else

{

ptr->next=temp->next;

free(temp);

}

}

display(head);

return head;

}

// searching in list

void search(struct node \*head)

{

int data,i=1;

struct node\* temp=head;

printf("Enter the data you want to search : ");

scanf("%d",&data);

while(temp!=NULL)

{

if(temp->data==data)

{

printf("found at %d node \n",i);

return;

}

temp=temp->next;

i++;

}

printf("not found");

}

// reversing the linked list

struct node\* reverse(struct node\* head)

{

struct node \*temp=NULL,\*ptr;

while(head)

{

ptr=head->next;

head->next=temp;

temp=head;

head=ptr;

}

display(temp);

return temp;

}

void main()

{

struct node\* head=NULL;

int choice;

while(1)

{

printf("\n");

printf("Enter the choice : \n");

printf("0. Exit\n");

printf("1. Display whole list\n");

printf("2. Insert at Beginning\n");

printf("3. Insert at End\n");

printf("4. Insert at Position\n");

printf("5. Delete from Beginning\n");

printf("6. Delete from End\n");

printf("7. Delete from Position\n");

printf("8. Search an Element\n");

printf("9. Reverse the linked List\n");

scanf("%d",&choice);

switch(choice)

{

case 0: exit(0);

break;

case 1: display(head);

break;

case 2: head = insertBeg(head);

break;

case 3: head = insertEnd(head);

break;

case 4: head = insertPos(head);

break;

case 5: head = deleteBeg(head);

break;

case 6: head = deleteEnd(head);

break;

case 7: head = deletePos(head);

break;

case 8: search(head);

break;

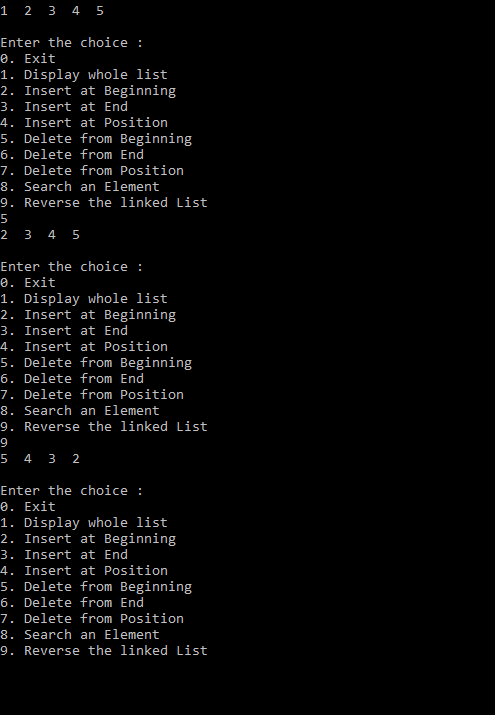
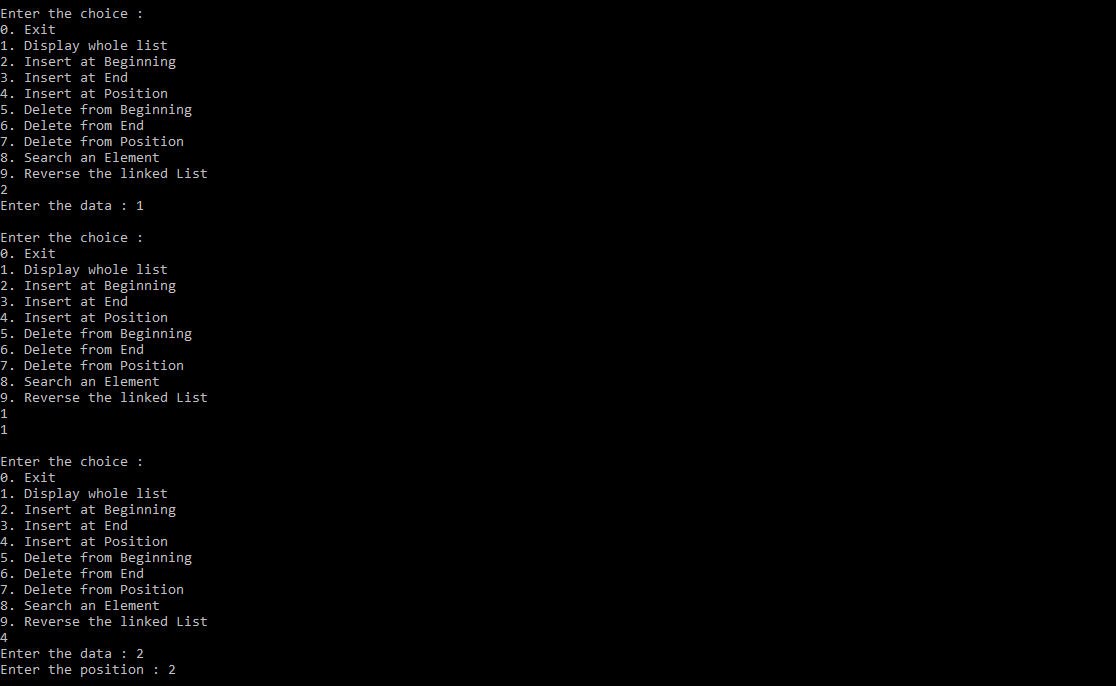
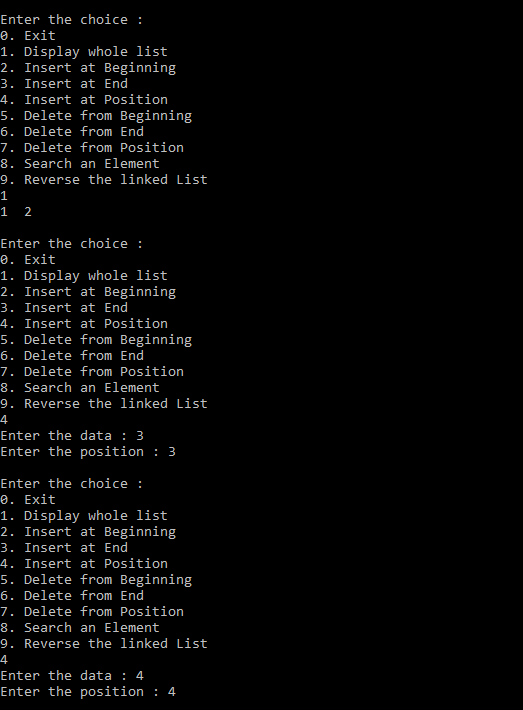
case 9: head=reverse(head);

break;

}

}

Output:



**Q 13. Write a C Program to implement Merge Sort**

#include<stdio.h>

void merge(int \*,int,int,int); //declaration of merge

void mergeSort(int \*arr,int low,int high)

{

int mid;

if(low>=high)

return;

mid=(low+high)/2;

mergeSort(arr,low,mid);

mergeSort(arr,mid+1,high);

merge(arr,low,mid,high);

}

void merge(int \*arr,int low,int mid,int high)

{

int i=low,j=mid+1,temp[100],pos=low,x;

while((i<mid+1)&&(j<=high))

{

if(arr[i]>arr[j])

{

temp[pos++]=arr[j];

j++;

}

else

{

temp[pos++]=arr[i];

i++;

}

}

// for remaining elements

while(i<mid+1)

{

temp[pos++]=arr[i];

i++;

}

while(j<=high)

{

temp[pos++]=arr[j];

j++;

}

for(x=low;x<=high;x++)

{

arr[x]=temp[x];

}

}

void main()

{

int n,arr[100],i;

printf("Enter size of array : ");

scanf("%d",&n);

printf("Enter elements of array : ");

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

{

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

}

mergeSort(arr,0,n-1);

printf("Sorted array : ");

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

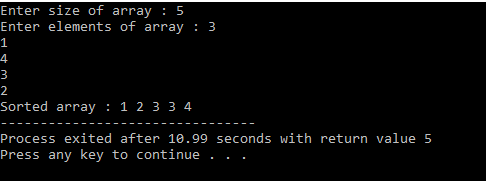
{

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

}

}

Output:



Q 14. Write a C Program to implement Quick Sort.

#include<stdio.h>

int findPivot(int \*arr,int low,int high)

{

int pivot=low,temp;

while(low<=high)

{

while(arr[low]<=arr[pivot])

low++;

while(arr[high]>arr[pivot])

high--;

if(low<high)

{

temp=arr[low];

arr[low]=arr[high];

arr[high]=temp;

}

}

// placing pivot to right position

temp=arr[pivot];

arr[pivot]=arr[high];

arr[high]=temp;

return high;

}

void quickSort(int \*arr,int low,int high)

{

if(low>=high)

return; //termination case

int pivot=findPivot(arr,low,high);

quickSort(arr,low,pivot-1);

quickSort(arr,pivot+1,high);

}

void main()

{

int n,arr[100],i;

printf("Enter size of array : ");

scanf("%d",&n);

printf("Enter elements of array : ");

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

{

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

}

quickSort(arr,0,n-1);

printf("Sorted array : ");

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

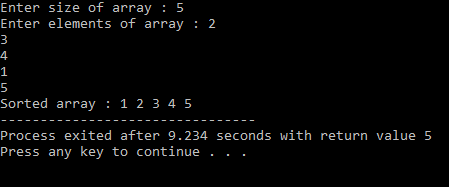
{

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

}

}

Output



**Q 15. Write a C Program to implement a Queue using two Stacks**

#include <stdio.h>

#include <stdlib.h>

/\* Functions and variables used \*/

void push1(int);

void push2(int);

int pop1();

int pop2();

void enqueue();

void dequeue();

void display();

void create();

int stack1[100], stack2[100];

int top1 = -1, top2 = -1;

int count = 0;

/\* Main Function \*/

int main()

{

int choice;

printf("\nQUEUE USING STACKS IMPLEMENTATION\n\n");

printf("\n1.ENQUEUE");

printf("\n2.DEQUEUE");

printf("\n3.DISPLAY");

printf("\n4.EXIT");

printf("\n");

create();

while (1)

{

printf("\nEnter your choice : ");

scanf("%d", &choice);

switch (choice)

{

case 1:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

exit(0);

default:

printf("\nInvalid Choice\n");

}}}

/\* Function to initialize top of two stacks\*/

void create()

{

top1 = top2 = -1;

}

/\* Function to push an element to stack \*/

void push1(int element)

{

stack1[++top1] = element; // Pushing the element to stack1

}

/\* Function to pop element from stack \*/

int pop1()

{

return(stack1[top1--]); // Pop element from stack1

}

/\* Function to push an element on to stack \*/

void push2(int element)

{

stack2[++top2] = element; // Pushing the element to stack2

}

/\* Function to pop an element from stack \*/

int pop2()

{

return(stack2[top2--]); // pop element from stack2

}

/\* Function to enqueue an element into the queue using stack \*/

void enqueue()

{

int data, i;

printf("Enter the data : ");

scanf("%d", &data);

push1(data); // Push data from stack to the queue

count++;

}

/\* Function to dequeue an element from the queue using stack \*/

void dequeue()

{

int i;

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

{

push2(pop1()); // Pop elements from stack1 and push them to stack2

}

pop2(); // Pop the element from stack2 which is the element to be dequeued

count--;

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

{

push1(pop2()); // Push back all the elements from stack2 to stack1

}}

/\*Function to display the elements in the queue\*/

void display()

{

int i;

if(top1 == -1)

{

printf("\nEMPTY QUEUE\n");

}

else

{

printf("\nQUEUE ELEMENTS : ");

for (i = 0;i <= top1;i++)

{

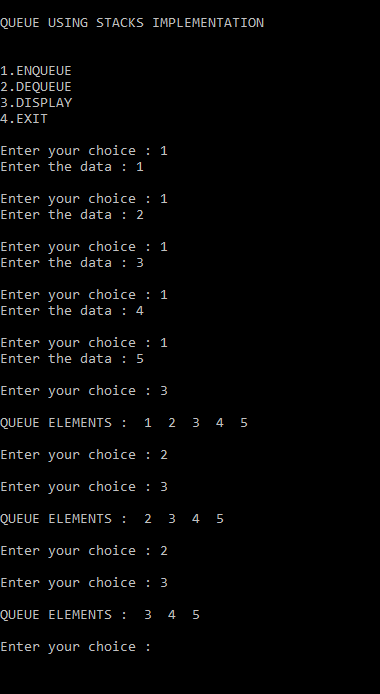
printf(" %d ", stack1[i]);

}

printf("\n");

}}

Output:



**Q 16. Write a C Program to**

1. **Preorder Traversal**
2. **Inorder Traversal**
3. **Postorder Traversal**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*right;

struct node \*left;

};

struct node\* insertData(struct node\* root)

{

int data;

struct node \*temp,\*prv;

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

printf("enter the data : ");

scanf("%d",&data);

newnode->data=data;

newnode->left=NULL;

newnode->right=NULL;

if(root==NULL)

{

root=newnode;

}

else

{

temp=root;

while(temp)

{

prv=temp;

if(data<temp->data)

{

temp=temp->left;

}

else

{

temp=temp->right;

}

}

if(data>prv->data)

prv->right=newnode;

else

prv->left=newnode;

}

return root;

}

//inorder traversal left->root->right

void inorder(struct node \*ptr)

{

if(ptr==NULL)

return;

inorder(ptr->left);

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

inorder(ptr->right);

}

//preorder traversal root->left->right

void preorder(struct node \*ptr)

{

if(ptr==NULL)

return;

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

preorder(ptr->left);

preorder(ptr->right);

}

//postorder traversal left->right->root

void postorder(struct node \*ptr)

{

if(ptr==NULL)

return;

postorder(ptr->left);

postorder(ptr->right);

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

}

int main()

{

struct node \*root=NULL;

while(1)

{

int ch;

printf("Enter your choice : \n");

printf("1. Insert Data : \n");

printf("2. Preorder traversal\n");

printf("3. Postorder traversal\n");

printf("4. Inorder traversal\n");

printf("5. Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 1: root=insertData(root);

break;

case 2: preorder(root);

printf("\n");

// break;

case 3: postorder(root);

printf("\n");

// break;

case 4: inorder(root);

printf("\n");

// break;

case 5: exit(0);

break;

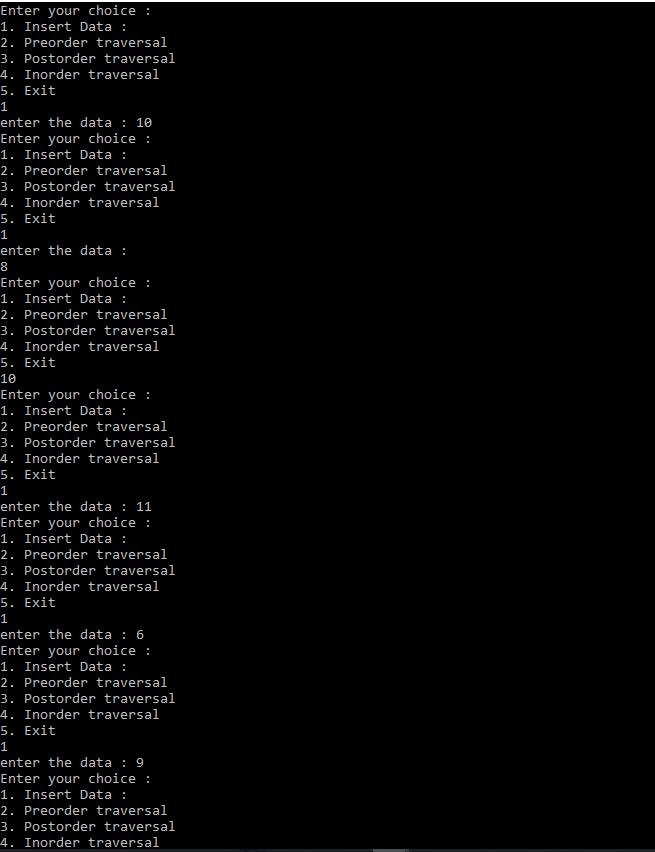
}

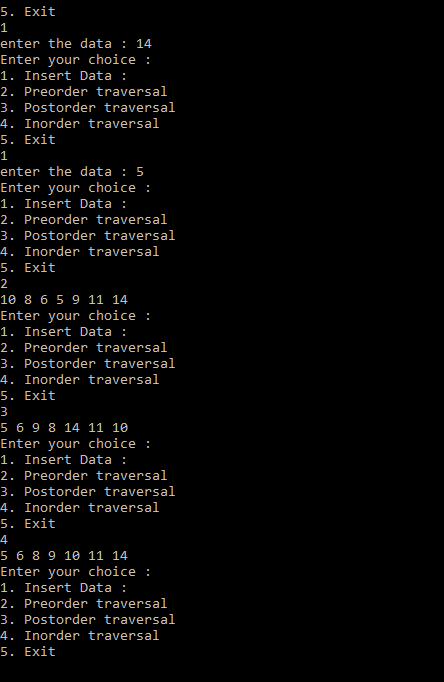
}

return 0;

}

**Output:**





**Q17. Write a C Program to implement BFS Traversal?**

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

#define initial 1

#define waiting 2

#define visited 3

int n;

int adj[MAX][MAX];

int state[MAX];

void create\_graph();

void BF\_Traversal();

void BFS(int v);

int queue[MAX], front = -1, rear = -1;

void insert\_queue(int vertex);

int delete\_queue();

int isEmpty\_queue();

int main()

{

create\_graph();

BF\_Traversal();

return 0;

}

void BF\_Traversal()

{

int v;

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

state[v] = initial;

printf("Enter Start Vertex for BFS: \n");

scanf("%d", &v);

BFS(v);

}

void BFS(int v)

{

int i;

insert\_queue(v);

state[v] = waiting;

while (!isEmpty\_queue())

{

v = delete\_queue();

printf("%d ", v);

state[v] = visited;

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

{

if (adj[v][i] == 1 && state[i] == initial)

{

insert\_queue(i);

state[i] = waiting;

}

}

}

printf("\n");

}

void insert\_queue(int vertex)

{

if (rear == MAX - 1)

printf("Queue Overflow\n");

else

{

if (front == -1)

front = 0;

rear = rear + 1;

queue[rear] = vertex;

}

}

int isEmpty\_queue()

{

if (front == -1 || front > rear)

return 1;

else

return 0;

}

int delete\_queue()

{

int delete\_item;

if (front == -1 || front > rear)

{

printf("Queue Underflow\n");

exit(1);

}

delete\_item = queue[front];

front = front + 1;

return delete\_item;

}

void create\_graph()

{

int count, max\_edge, origin, destin;

printf("Enter number of vertices : ");

scanf("%d", &n);

max\_edge = n \* (n - 1);

for (count = 1; count <= max\_edge; count++)

{

printf("Enter edge %d( -1 -1 to quit ) : ", count);

scanf("%d %d", &origin, &destin);

if ((origin == -1) && (destin == -1))

break;

if (origin >= n || destin >= n || origin < 0 || destin < 0)

{

printf("Invalid edge!\n");

count--;

}

else

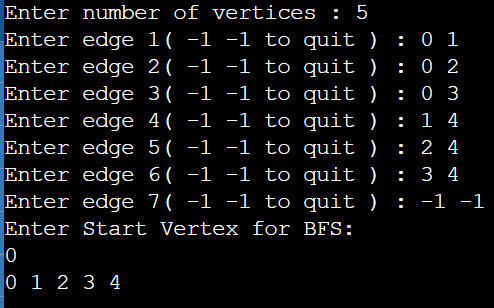
{

adj[origin][destin] = 1;

}

}

}



**Q18. Write a C Program to implement DFS Traversal?**

#include <stdio.h>

void DFS(int);

int G[10][10], visited[10], n; // n is no of vertices and graph is sorted in array G[10][10]

void main()

{

int i, j;

printf("Enter number of vertices:");

scanf("%d", &n);

// read the adjecency matrix

printf("\nEnter adjecency matrix of the graph:");

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

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

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

// visited is initialized to zero

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

visited[i] = 0;

DFS(0);

}

void DFS(int i)

{

int j;

printf("\n%d", i);

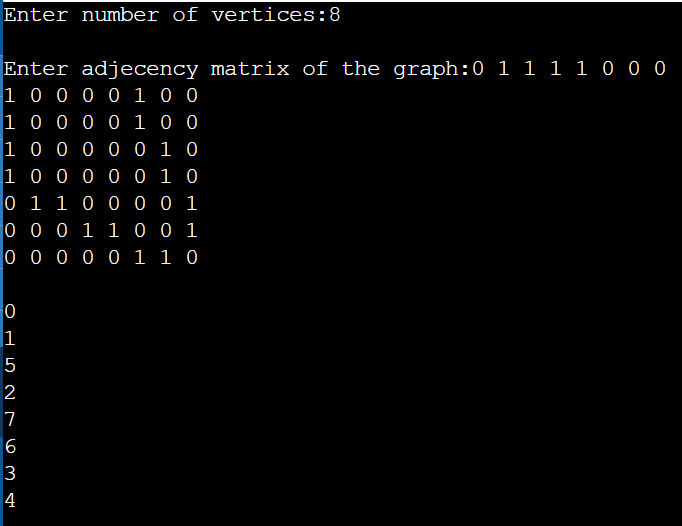
visited[i] = 1;

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

if (!visited[j] && G[i][j] == 1)

DFS(j);

}



Q19. Write a C Program to implement Prim’s Algorithm?

#include <stdio.h>

#include <stdlib.h>

#define infinity 9999

#define MAX 20

int G[MAX][MAX], spanning[MAX][MAX], n;

int prims();

int main()

{

int i, j, total\_cost;

printf("Enter no. of vertices:");

scanf("%d", &n);

printf("\nEnter the adjacency matrix:\n");

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

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

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

total\_cost = prims();

printf("\nspanning tree matrix:\n");

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

{

printf("\n");

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

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

}

printf("\n\nTotal cost of spanning tree=%d", total\_cost);

return 0;

}

int prims()

{

int cost[MAX][MAX];

int u, v, min\_distance, distance[MAX], from[MAX];

int visited[MAX], no\_of\_edges, i, min\_cost, j;

// create cost[][] matrix,spanning[][]

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

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

{

if (G[i][j] == 0)

cost[i][j] = infinity;

else

cost[i][j] = G[i][j];

spanning[i][j] = 0;

}

// initialise visited[],distance[] and from[]

distance[0] = 0;

visited[0] = 1;

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

{

distance[i] = cost[0][i];

from[i] = 0;

visited[i] = 0;

}

min\_cost = 0; // cost of spanning tree

no\_of\_edges = n - 1; // no. of edges to be added

while (no\_of\_edges > 0)

{

// find the vertex at minimum distance from the tree

min\_distance = infinity;

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

if (visited[i] == 0 && distance[i] < min\_distance)

{

v = i;

min\_distance = distance[i];

}

u = from[v];

// insert the edge in spanning tree

spanning[u][v] = distance[v];

spanning[v][u] = distance[v];

no\_of\_edges--;

visited[v] = 1;

// updated the distance[] array

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

if (visited[i] == 0 && cost[i][v] < distance[i])

{

distance[i] = cost[i][v];

from[i] = v;

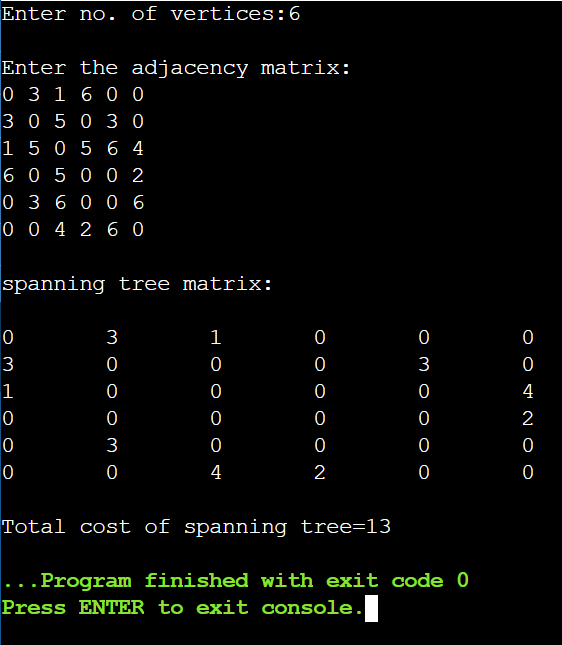
}

min\_cost = min\_cost + cost[u][v];

}

return (min\_cost);

}



Q20. Write a C Program to implement Kruskal’s Algorithm?

#include <stdio.h>

#define MAX 30

typedef struct edge

{

int u, v, w;

} edge;

typedef struct edgelist

{

edge data[MAX];

int n;

} edgelist;

edgelist elist;

int G[MAX][MAX], n;

edgelist spanlist;

void kruskal();

int find(int belongs[], int vertexno);

void union1(int belongs[], int c1, int c2);

void sort();

void print();

void main()

{

int i, j, total\_cost;

printf("\nEnter number of vertices:");

scanf("%d", &n);

printf("\nEnter the adjacency matrix:\n");

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

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

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

kruskal();

print();

}

void kruskal()

{

int belongs[MAX], i, j, cno1, cno2;

elist.n = 0;

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

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

{

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

{

elist.data[elist.n].u = i;

elist.data[elist.n].v = j;

elist.data[elist.n].w = G[i][j];

elist.n++;

}

}

sort();

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

belongs[i] = i;

spanlist.n = 0;

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

{

cno1 = find(belongs, elist.data[i].u);

cno2 = find(belongs, elist.data[i].v);

if (cno1 != cno2)

{

spanlist.data[spanlist.n] = elist.data[i];

spanlist.n = spanlist.n + 1;

union1(belongs, cno1, cno2);

}

}

}

int find(int belongs[], int vertexno)

{

return (belongs[vertexno]);

}

void union1(int belongs[], int c1, int c2)

{

int i;

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

if (belongs[i] == c2)

belongs[i] = c1;

}

void sort()

{

int i, j;

edge temp;

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

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

if (elist.data[j].w > elist.data[j + 1].w)

{

temp = elist.data[j];

elist.data[j] = elist.data[j + 1];

elist.data[j + 1] = temp;

}

}

void print()

{

int i, cost = 0;

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

{

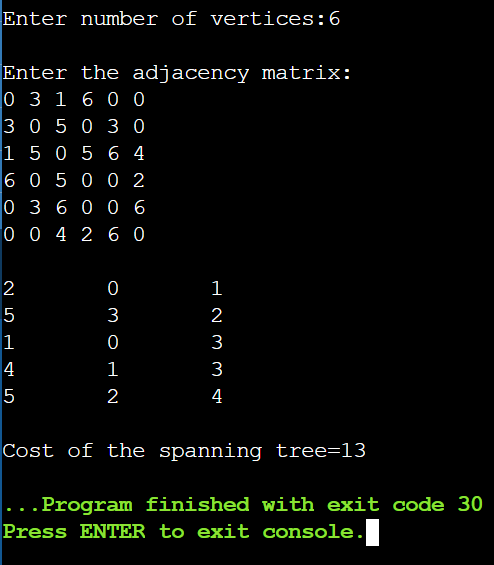
printf("\n%d\t%d\t%d", spanlist.data[i].u, spanlist.data[i].v, spanlist.data[i].w);

cost = cost + spanlist.data[i].w;

}

printf("\n\nCost of the spanning tree=%d", cost);

}



Q21. Write a C Program to read data from a file?

#include <stdio.h>

#include <string.h>

int main()

{

FILE \*filePointer;

char dataToBeRead[50];

filePointer = fopen("temp.txt", "r");

if (filePointer == NULL)

{

printf("temp.txt file failed to open.");

}

else

{

printf("The file is now opened.\n");

while (fgets(dataToBeRead, 50, filePointer) != NULL)

{

printf("%s", dataToBeRead);

}

fclose(filePointer);

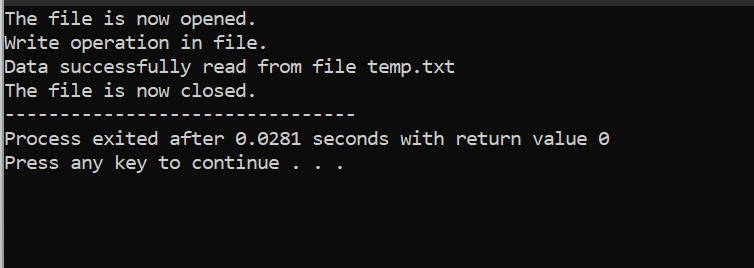
printf("Data successfully read from file GfgTest.c\n");

printf("The file is now closed.");

}

return 0;

}



Q22. Write a C Program to write data in a file?

#include <stdio.h>

#include <string.h>

int main()

{

FILE \*filePointer;

char dataToBeWritten[50] = "Write operation in file.";

filePointer = fopen("temp.txt", "w");

if (filePointer == NULL)

{

printf("temp.txt file failed to open.");

}

else

{

printf("The file is now opened.\n");

if (strlen(dataToBeWritten) > 0)

{

fputs(dataToBeWritten, filePointer);

fputs("\n", filePointer);

}

fclose(filePointer);

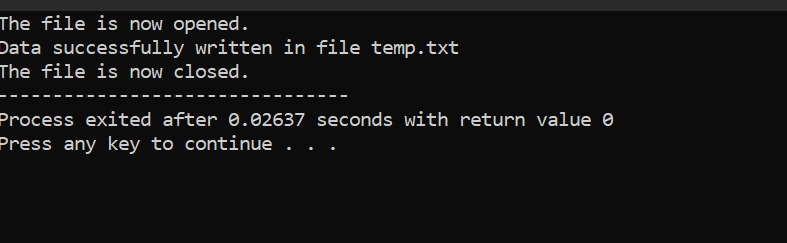
printf("Data successfully written in file temp.txt\n");

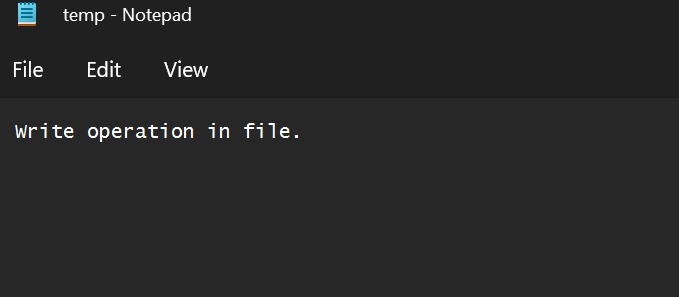
printf("The file is now closed.");

}

return 0;

}





Q23. Write a C Program to replace all occurrences of a given word with another word in a file?

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void findAndReplaceInFile()

{

FILE \*ifp, \*ofp;

char word[100], ch, read[100], replace[100];

int word\_len, i, p = 0;

ifp = fopen("temp.txt", "r");

ofp = fopen("output.txt", "w+");

if (ifp == NULL || ofp == NULL)

{

printf("Can't open file.");

exit(0);

}

puts("THE CONTENTS OF THE FILE ARE SHOWN BELOW :\n");

while (1)

{

ch = fgetc(ifp);

if (ch == EOF)

{

break;

}

printf("%c", ch);

}

puts("\n\nEnter the word to find:");

fgets(word, 100, stdin);

word[strlen(word) - 1] = word[strlen(word)];

puts("Enter the word to replace it with :");

fgets(replace, 100, stdin);

replace[strlen(replace) - 1] = replace[strlen(replace)];

fprintf(ofp, "%s - %s\n", word, replace);

rewind(ifp);

while (!feof(ifp))

{

fscanf(ifp, "%s", read);

if (strcmp(read, word) == 0)

{

strcpy(read, replace);

}

fprintf(ofp, "%s ", read);

}

rewind(ofp);

while (1)

{

ch = fgetc(ofp);

if (ch == EOF)

{

break;

}

printf("%c", ch);

}

fclose(ifp);

fclose(ofp);

}

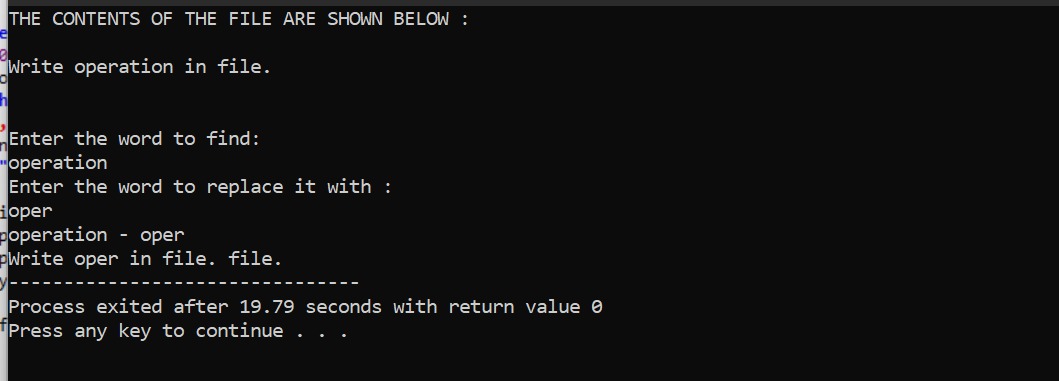
int main()

{

findAndReplaceInFile();

return 0;

}



Q24. Write a C Program to implement Floyd-Warshall Algorithm?

WAP to implement Floyd Warshall Algorithm in C

#define MAXSIZE 10

#define INF 1000

#include <stdio.h>

#include <stdlib.h>

int main()

{

int i, j, k; // for looping

int n; // number of nodes.

int adj[MAXSIZE][MAXSIZE];

printf("Enter the number of nodes: ");

scanf("%d", &n);

// initialize cost matrix adj

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

{

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

{

if (i == j)

{

adj[i][j] = 0;

}

else

{

adj[i][j] = INF;

}

}

}

// taking input for edges

printf("Enter node1, node2 and edge-weight(-1 -1 -1 to stop): \n");

int inLoop = 1;

while (inLoop == 1)

{

int v1, v2, w;

scanf("%d %d %d", &v1, &v2, &w);

if (v1 > -1 && v2 > -1)

{

// register this edge in cost matrix

adj[v1][v2] = w;

}

else

{

inLoop = 0;

}

}

printf("adj matrix is as follows\n");

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

{

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

{

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

}

printf("\n");

}

printf("\n");

// applying floyd warshall algorithm

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

{

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

{

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

{

if (i == j)

continue;

if (adj[i][j] > adj[i][k] + adj[k][j])

{

adj[i][j] = adj[i][k] + adj[k][j];

}

}

}

}

printf("All pairs shortest path matrix is as follows.\n");

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

{

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

{

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

}

printf("\n");

}

}

