

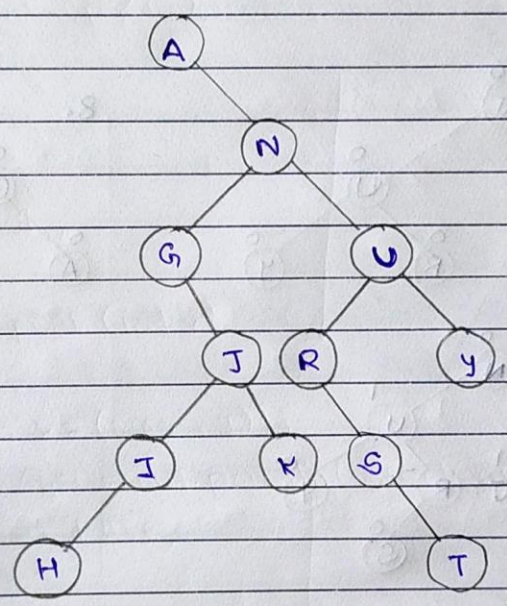
Name: Anurag. A. Thakur
 Roll.No: 205
 PRN No: 2046491246005

Q1. Create a BST and AVL Tree for your own full name
 Sol:-

ANURAG AJAYJINGH THAKUR.

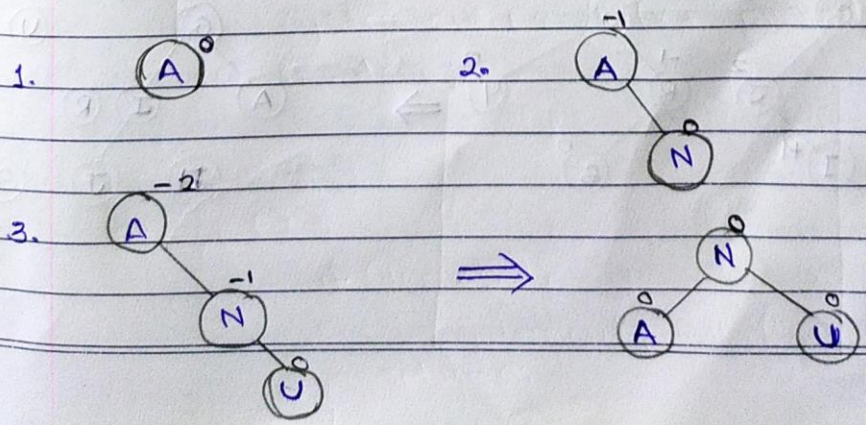
ANURAGJYSGJHTK.

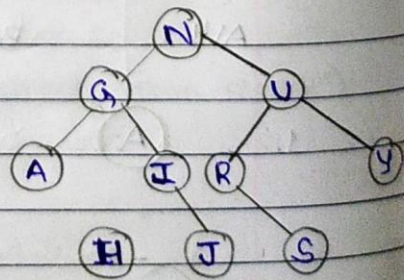
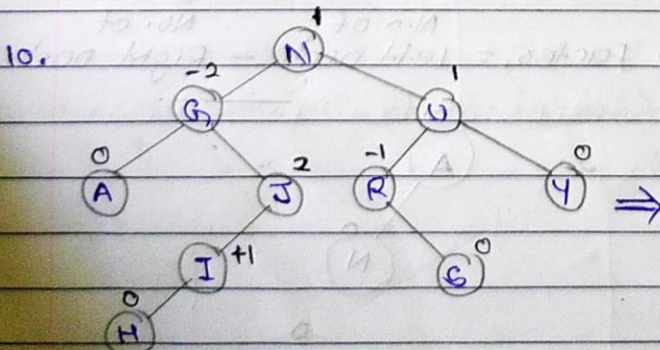
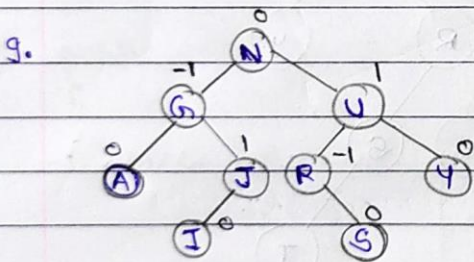
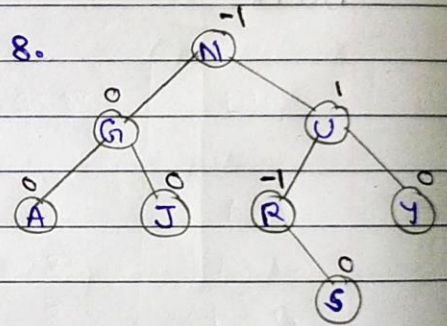
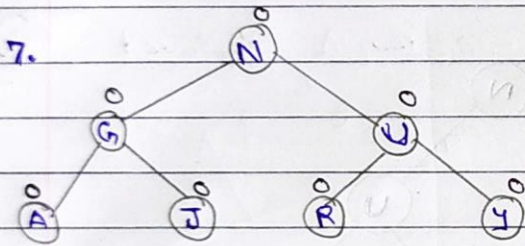
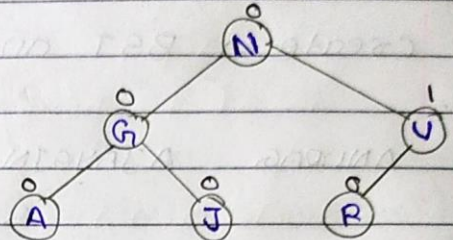
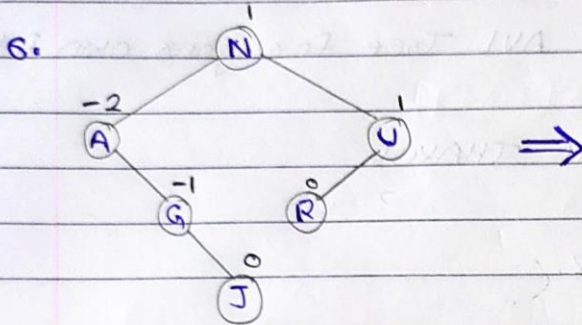
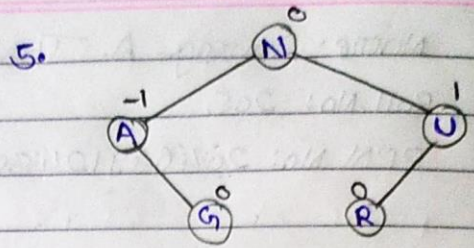
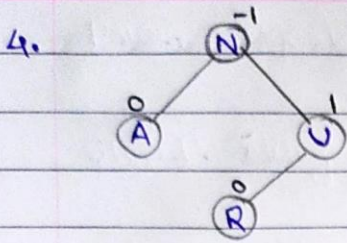
BST:-

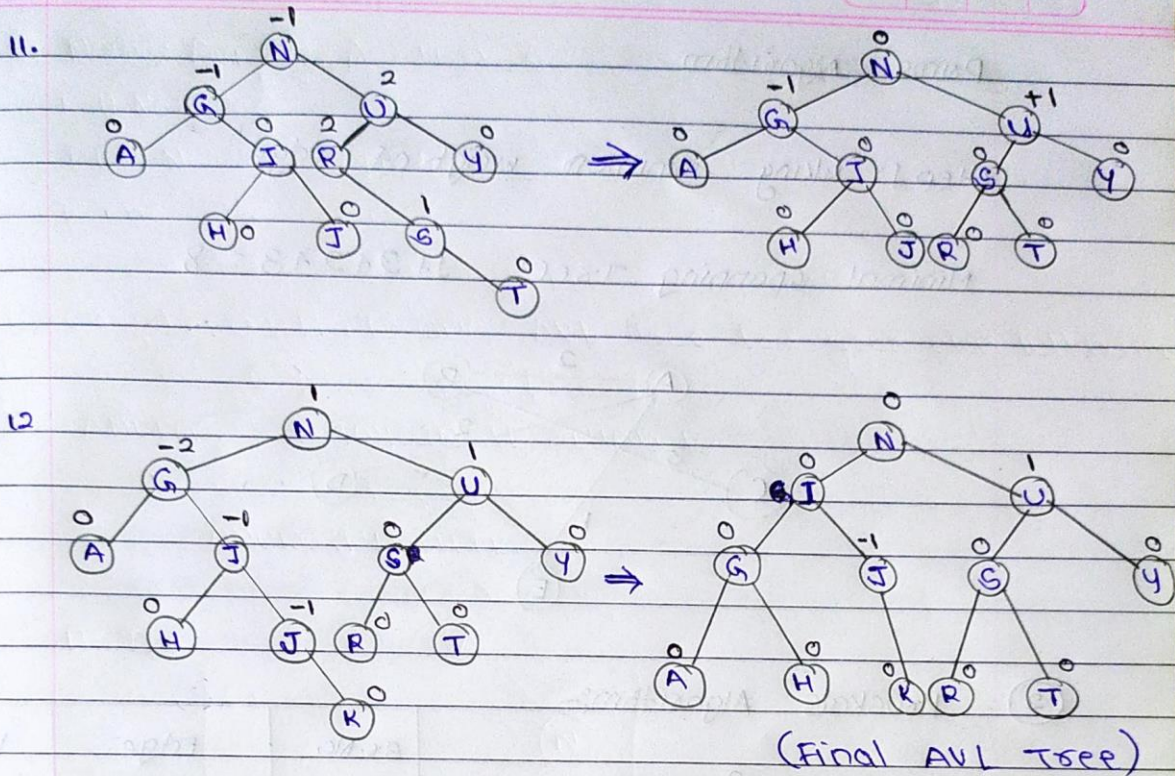


AVL:

Balance Factor = $\frac{\text{No. of Left node} - \text{No. of Right node}}{2}$



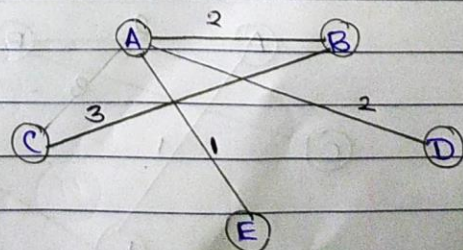
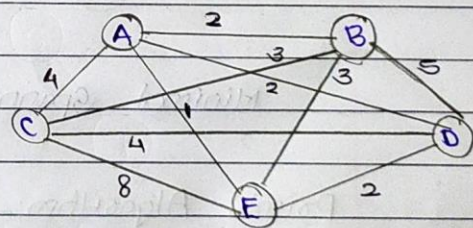




Q4. For the given graph, calculate minimal spanning tree using Prim's and Kruskal's Algorithm

①

Soln:	S.S. No	Edge	Weight
	1.	A-E	1
	2.	A-B	2
	3.	A-D	2
	4.	D-E	2
	5.	B-C	3
	6.	B-E	3
	7.	A-C	4
	8.	C-D	4
	9.	B-D	5
	10.	E-C	8

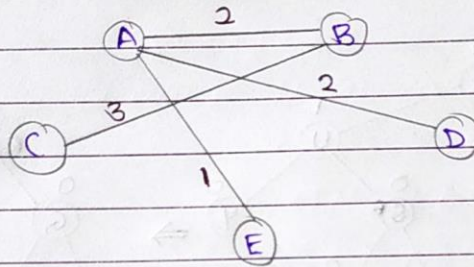


Kruskal's Algorithm = $2+2+1+3$
= 8

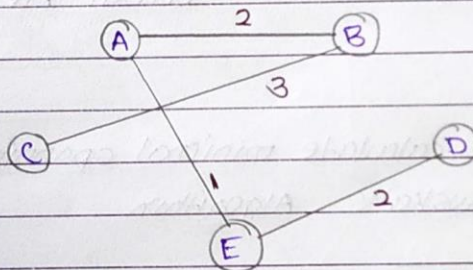
Prims Algorithm:-

Step 1: Taking minimum weight of edge.

Minimal Spanning Tree: $1+2+2+3=8$



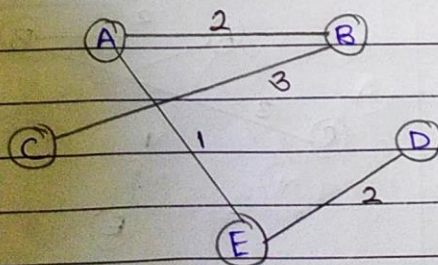
(2) Kruskals Algorithm:-



Sr. No	Edge	Weight
1.	A-E	1
2.	A-B	2
3.	E-D	2
4.	B-C	3

Minimal Spanning Tree = $1+2+2+3=8$

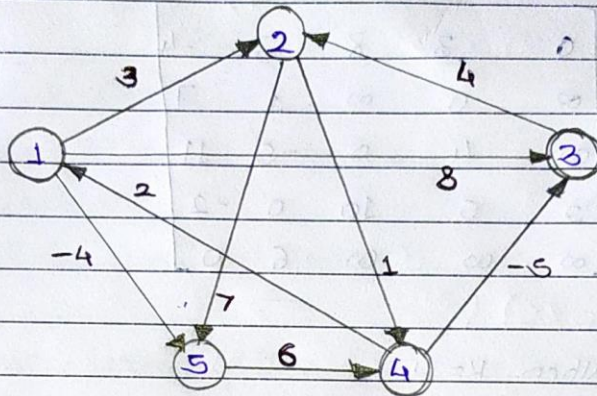
Prims Algorithm:-



Minimal Spanning Tree-

$1+2+2+3=8$

Q5. Find all pairs shortest path using Floyd - Warshall Algorithm



Soln:-

Step 1: $K=0$, $dij^{(K)} = \min(dij^{(K-1)}, dij^{(K-1)} + dkj^{(K-1)})$

$$D(0) = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & -5 & \infty \\ 2 & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Step 2: $K=1$

$$D(1) = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & -5 & \infty \\ 2 & 5 & 10 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Step 3: $K=2$

$$D(2) = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & -5 & 11 \\ 2 & 5 & 10 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Step 4: When $K=3$

$$D(3) = \begin{bmatrix} 0 & 3 & 8 & 3 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & -5 & 11 \\ 2 & 5 & 10 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

Step 5: When $K=4$

$$D(4) = \begin{bmatrix} 0 & 3 & 8 & 3 & -4 \\ 3 & 0 & 11 & 1 & -1 \\ -3 & 0 & 0 & -5 & -7 \\ 2 & 5 & 10 & 0 & -2 \\ 8 & 11 & 16 & 6 & 0 \end{bmatrix}$$

Step 6: When $K=5$

$$D(5) = \begin{bmatrix} 0 & 3 & 8 & 3 & -4 \\ 3 & 0 & 11 & 1 & -1 \\ -3 & 0 & 0 & -5 & -7 \\ 2 & 5 & 10 & 0 & -2 \\ 8 & 11 & 16 & 6 & 0 \end{bmatrix}$$

Q2. Write a non recursive C program to implement inorder , preorder and postorder traversal for a BST.

PROGRAM:

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

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

```
#define MAX 50
```

```
struct
```

```
{ struct node
```

```
*left;
```

```
int info; struct
```

```
node *right;
```

```
};
```

```
struct node *insert(struct node *root, int ele );
```

```
void preorder(struct node *root); void
```

```
inorder(struct node *root); void
```

```
postorder(struct node *root); void
```

```
display(struct node *ptr,int level);
```

```
struct node *queue[MAX];
```

```
int front=-1,rear=-1; void
```

```
enqueue(struct node *item);
```

```
struct node *DeQue(); int
```

```
Qempty();
```

```
struct node *stack[MAX]; int
```

```
top=-1; void Push(struct
```

```
node *item); struct node
```

```
*Pop(); int StkEmt();
```

```
int main( )
```

```
{
```

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005

```

    struct node *root=NULL;

    int x,y, ele;
while(y=1)
    {
        printf("\n 1.Insert");

printf("\n 2.Display");
printf("\n 3.Preorder");
printf("\n 4.Inorder");
printf("\n 5.Postorder");
printf("\n 6.Exit ");
printf("\n Choose option:");
scanf("%d",&x);        switch(x)
    {
case 1:
        printf("\nEnter Element : ");
scanf("%d",&ele);        root =
insert(root, ele);

        break;

        case 2:
            printf("\n\t");
display(root,0);
printf("\n\t");        break;
        case
3:
            preorder(root);
            break;

        case 4:
inorder(root);
break;

```

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005

```
        case 5:
postorder(root);
        break;
        case 6:
exit(1);
y=0;

        default:
            printf("\nWrong Input!!!x\n");
        }
    }
    return 0;
}

struct node *insert(struct node *root, int ele)
{
    struct node *tmp,*p,*ptr;
    ptr = root;    p
= NULL;    while(
ptr!=NULL)
    {
        p
= ptr;
        if(ele < ptr->info) ptr
            = ptr->left;

        else
ptr = ptr->right;
    }
    tmp=(struct node *)malloc(sizeof(struct node));
    tmp->info=ele;    tmp->left=NULL;    tmp-
```



```

>right=NULL;    if(p==NULL)        root=tmp;
else if( ele < p->info )        p->left=tmp;    else
p->right=tmp;    return root;
}

```

```

void preorder(struct node *root)

```

```

{
    struct node *ptr = root;
    if( ptr==NULL )
    {
        printf("Tree is empty\n");
    }
    return;
}

```

```

    printf("\n Pre-order : ");

```

```

    Push(ptr);        while(

```

```

    !StkEmt() )

```

```

    {
        ptr = Pop();
        printf("%d ",ptr->info);
        if(ptr->right!=NULL)
            Push(ptr->right);
        if(ptr->left!=NULL)
            Push(ptr->left);
    }

```

```

    printf("\t");

```

```

}

```

```

void inorder(struct node *root)

```

```

{

```

```

    struct node *ptr=root;

```

```

    if( ptr==NULL )

```

```

    {

```

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005


```

        printf("Tree is empty\n");
        return;
    }
    printf("\n In-order : ");
while(1)
    {
        while(ptr->left!=NULL )
            {
                Push(ptr);
                ptr = ptr->left;
            }
        while( ptr->right==NULL )
            {
                printf("%d ",ptr->info);
                if(StkEmt()) return;
                ptr = Pop();
            }
        printf("%d ",ptr->info);
        ptr = ptr->right;
    }
}

void postorder(struct node *root)
{
    struct node *ptr = root;
    struct node *q;    if(
    ptr==NULL )
    {
        printf("Tree is empty\n");
    }
    return;
}

```

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005


```

        q = root;        printf("\n
Post-order : ");        while(1)
    {
        while(ptr->left!=NULL)
        {
            Push(ptr);
ptr=ptr->left;
        }
        while( ptr->right==NULL || ptr->right==q )
        {
            printf("%d ",ptr->info);
q = ptr;                if( StkEmt() )
return;                ptr = Pop();
        }
        Push(ptr);
ptr = ptr->right;
    }
    printf("\t");
}

void enqueue(struct node *item)
{
    if(rear==MAX-1)
    {
        printf("queue Overflow\n");
        return;
    }
    if(front== -1)
front=0;
rear=rear+1;
queue[rear]=item ;
}

```

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005

```

struct node *DeQue()
{
    struct node *item;
    if(front==-1 || front==rear+1)
    {
        printf("queue Underflow\n");
        return 0;
    }
    item=queue[front];
    front=front+1;    return
    item;
}
int Qempty()
{
    if(front==-1 || front==rear+1)
        return 1;
    else
        return 0;
}
void Push(struct node *item)
{
    if(top==(MAX-1))
    {
        printf("stack Overflow\n");
        return;
    }
    top=top+1;
    stack[top]=item;
}
struct node *Pop()
{

```

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005


```

    struct node *item;
if(top== -1)
    {
        printf("stack Underflow....\n");
        exit(1);
    }
    item=stack[top];
top=top-1;    return
item;
}
int StkEmt()
{
    if(top== -1)
return 1;    else
return 0;
}
void display(struct node *ptr,int level)
{    int
i;
    if(ptr == NULL )
return;    else
    {
        display(ptr->right, level+1);
printf("\n\t");        for (i=0;
i<level; i++)
        printf(" ");
printf("%d", ptr->info);
display(ptr->left, level+1);
    }
}  OUTPUT:-

```

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005

```
1.Insert
2.Display
3.Preorder
4.Inorder
5.Postorder
6.Exit
Choose option:1
```

Enter Element : 50

```
1.Insert
2.Display
3.Preorder
4.Inorder
5.Postorder
6.Exit
Choose option:1
```

Enter Element : 60

```
1.Insert
2.Display
3.Preorder
4.Inorder
5.Postorder
6.Exit
Choose option:1
```

Enter Element : 40

```
1.Insert
2.Display
3.Preorder
4.Inorder
5.Postorder
6.Exit
Choose option:1
```


NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005

Enter Element : 35

1.Insert
2.Display
3.Preorder
4.Inorder
5.Postorder
6.Exit
Choose option:1

Enter Element : 55

1.Insert
2.Display
3.Preorder
4.Inorder
5.Postorder
6.Exit
Choose option:1

Enter Element : 70

1.Insert
2.Display
3.Preorder
4.Inorder
5.Postorder
6.Exit
Choose option:2

```

              70
            60
          55
        50
      40
    35
```

1.Insert
2.Display
3.Preorder
4.Inorder
5.Postorder
6.Exit
Choose option:3

Pre-order : 50 40 35 60 55 70

1.Insert
2.Display
3.Preorder
4.Inorder
5.Postorder
6.Exit
Choose option:4

In-order : 35 40 50 55 60 70

1.Insert
2.Display
3.Preorder
4.Inorder
5.Postorder
6.Exit
Choose option:5

Post-order : 35 40 55 70 60 50

1.Insert
2.Display
3.Preorder
4.Inorder
5.Postorder
6.Exit
Choose option:6

...Program finished with exit code 0
Press ENTER to exit console.

Q3. Write C program to implement Depth first search and Breadth first search traversals of a graph.

BFS PROGRAM:

```
#include<stdio.h> #include<stdlib.h> int  
a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;
```

```
void bfs(int v) { for(i =  
1; i <= n; i++) if(a[v][i]  
&& !visited[i]) q[++r] =  
i; if(f <= r) {  
visited[q[f]] = 1;  
bfs(q[f++]);  
}  
}
```

```
void main() {  
int v; printf("\n Enter the number of  
vertices:"); scanf("%d", &n);
```

```
for(i=1; i <= n; i++) {  
q[i] = 0; visited[i] =  
0;  
}
```

```
printf("\n Enter graph data in matrix form:\n");  
for(i=1; i<=n; i++) { for(j=1; j<=n; j++) {  
printf("Enter the number for a[%d][%d]:", i, j);  
scanf("%d", &a[i][j]);  
}  
}
```

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005


```
printf("\n Enter the starting vertex:");  
scanf("%d", &v);  
bfs(v); printf("\n The node which are reachable  
are:\n");
```

```
for(i=1; i <= n; i++) {  
if(visited[i])  
printf("%d\t", i); else  
{  
printf("\n Bfs is not possible. Not all nodes are reachable");  
break;  
}  
}  
} OUTPUT:
```

```
Enter the number of vertices:4  
  
Enter graph data in matrix form:  
Enter the number for a[1][1]:1  
Enter the number for a[1][2]:2  
Enter the number for a[1][3]:3  
Enter the number for a[1][4]:4  
Enter the number for a[2][1]:5  
Enter the number for a[2][2]:6  
Enter the number for a[2][3]:3  
Enter the number for a[2][4]:2  
Enter the number for a[3][1]:1  
Enter the number for a[3][2]:6  
Enter the number for a[3][3]:5  
Enter the number for a[3][4]:4  
Enter the number for a[4][1]:1  
Enter the number for a[4][2]:5  
Enter the number for a[4][3]:3  
Enter the number for a[4][4]:6  
  
Enter the starting vertex:3  
  
The node which are reachable are:  
1      2      3      4  
  
...Program finished with exit code 0  
Press ENTER to exit console.□
```

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005

DFS PROGRAM:

```
include<stdio.h>

#include<conio.h> int
a[20][20],reach[20],n;

void dfs(int v)

{ int
i;

reach[v]=1;

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

if(a[v][i] && !reach[i])

{

printf("\n %d->%d",v,i);

dfs(i);

}

}

void main()

{

int i,j,count=0; printf("\n Enter
number of vertices:");

scanf("%d",&n); for(i=1;i<=n;i++)

{

reach[i]=0;

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

a[i][j]=0;

}

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

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

scanf("%d",&a[i][j]); dfs(1);

printf("\n");

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

{
```

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005

```
    if(reach[i])
count++;
}
if(count==n) printf("\n Graph is
connected");
else printf("\n Graph is not
connected");
}
```

NAME:ANURANG A THAKUR

ROLL NO.:205

PRN NO.:2046491245005

OUTPUT:

```
Enter number of vertices:4
Enter the adjacency matrix:
1
2
3
4
5
6
3
2
1
4
8
6
7
2
0
1

1->2
2->3
3->4

Graph is connected

...Program finished with exit code 0
Press ENTER to exit console.
```