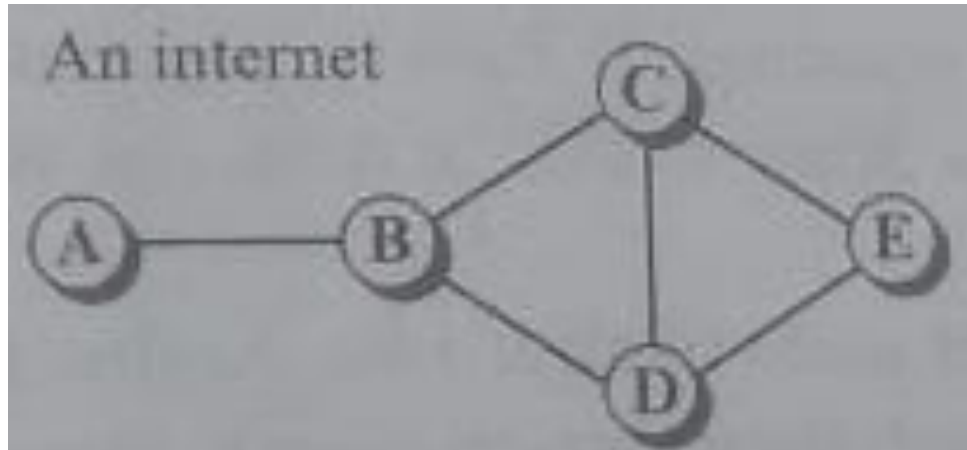


Bijay REGMI (210913032)

WEEK - 2



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

#define INFINITY 999

int n;
int c[100][100];

void inputMatrix(){
    printf("\nEnter the adjacency matrix : \n");
    int i,j;
    for(i=0; i<n; i++){
        for(j=0; j<n; j++){
            scanf("%d",&c[i][j]);
        }
    }
}

void displayMatrix(){
    int i,j;
    printf("\nAdjacent Matrix : \n");

    for(i=0; i<n; i++){
        for(j=0; j<n; j++){
            printf("    %d",c[i][j]);
```

```

    }
    printf("\n");
}
}

void PVR(int startnode,int destnode, int exception){

    int distance[100],pred[100];
    int visited[100],count,mindistance,nextnode,i,j;

    // //initialize pred[],distance[] and visited[]
    for(i=0;i<n;i++)
    {
        distance[i]=c[startnode][i];
        pred[i]=startnode;
        visited[i]=0;
    }
    distance[startnode]=0;
    visited[startnode]=1;
    count=1;
    while(count<n-1)
    {
        mindistance=999;
        //nextnode gives the node at minimum distance
        for(i=0;i<n;i++){
            if(distance[i]<mindistance&&!visited[i])
            {
                mindistance=distance[i];
                nextnode=i;
            }
        }
        //check if a better path exists through nextnode
        visited[nextnode]=1;
        for(i=0;i<n;i++){
            if(!visited[i]){
                if(mindistance+c[nextnode][i]<distance[i])
                {
                    distance[i]=mindistance+c[nextnode][i];
                    pred[i]=nextnode;
                }
            }
        }
        count++;
    }
    int e;
    //print the path and distance of each node
    for(i=startnode;i<=destnode;i++){

```

```

        // If exception is there
        if(i!=startnode && i!=exception)
        {
            printf("\nPath to %c : %c", i+65, i+65);
            j=i;
            while(j!=startnode)
            {
                j=pred[j];
                printf("<-%c",j+65);
            }
        }
        printf("\n");
    }
}

```

```

int main(){
    int source,destn,exception;
    char s,d,e;

    printf("Enter the number of nodes : ");
    scanf("%d",&n);

    inputMatrix();
    displayMatrix();

    printf("\nEnter the source Node : ");
    scanf(" %c",&s);
    printf("\nEnter the destination Node : ");
    scanf(" %c",&d);

    source = ((int)toupper(s))-65;
    destn = ((int)toupper(d))-65;

    // Path Vector Routing
    PVR(source, destn, -1);

    printf("\n\nEnter the Node not to be visited : ");
    scanf(" %c",&e);
    exception= ((int)toupper(e))-65;

    // Deleting row and column
    int i;
    for(i=0;i<n;i++){
        c[i][exception] = 999;
        c[exception][i] = 999;
    }
}

```

```
// Path Vector Routing
PVR(source, destn, exception);

return 0;
}
```

```

Last login: Mon Nov 29 18:14:03 on ttys005
regmi@Bijays-MacBook-Air ~ % cd Documents/ACNW
regmi@Bijays-MacBook-Air ACNW % ls
BGP.c          BGP_Lab.c      BGP_Sandhya.c  a.out
regmi@Bijays-MacBook-Air ACNW % gcc BGP_Lab.c
regmi@Bijays-MacBook-Air ACNW % ./a.out
Enter the number of nodes : 5

Enter the adjacency matrix :
0 1 999 999 999
1 0 1 1 999
999 1 0 999 1
999 1 999 0 1
999 999 1 1 0

Adjacent Matrix :
      0      1      999      999      999
      1      0      1      1      999
      999      1      0      999      1
      999      1      999      0      1
      999      999      1      1      0

Enter the source Node : a

Enter the destination Node : e

Path to B : B<-A
Path to C : C<-B<-A
Path to D : D<-B<-A
Path to E : E<-C<-B<-A

Enter the Node not to be visited : c

Path to B : B<-A
Path to D : D<-B<-A
Path to E : E<-D<-B<-A
regmi@Bijays-MacBook-Air ACNW %

```