**PRIMS**

```
#include<stdio.h>
#define  mx  99
int w[5][5]={ {0,10,5,mx,mx},
                                {10, 0,2,8,3},
                                {5,2,0,mx,5},
                                {mx,8,mx,0,1},
                                {mx,3,5,1,0} };
struct vertexnode
        {
        int vertex;
        int key;
        int p;
        };
int qsize=0;
struct vertexnode V[5];
void QInsert(struct vertexnode Q[],struct vertexnode V[],int n);
int inQueue(struct vertexnode Q[],struct vertexnode v);
void MSTPrim(struct vertexnode V[],int w[][5],int n,int s);
struct vertexnode extract_min(struct vertexnode Q[]);
void printV(struct vertexnode V[],int n);
void sort(struct vertexnode Q[],int n);
void printQ(struct vertexnode V[],int n);
int main()
        {
        int i,n=5;
        for(i=0;i<n;i++)
                {
                V[i].vertex =i;
                }
        MSTPrim(V,w,5,0);
        printV(V,n);

        }
void printV(struct vertexnode V[],int n)
        {
        int i;
        printf("\n\nvertex\t:");
        for(i=0;i<n;i++)
                printf(" %5d",V[i].vertex);
        printf("\nKeys\t:");
        for(i=0;i<n;i++)
                printf(" %5d",V[i].key);
        printf("\nParent\t:");
        for(i=0;i<n;i++)
                printf(" %5d",V[i].p);
        }
void printQ(struct vertexnode V[],int n)
```

```
	{
	int i;
	printf("\n\nQueue  :");
	for(i=0;i<n;i++)
		printf(" %5d",V[i].vertex);
	}
void MSTPrim(struct vertexnode V[],int w[][5],int n,int s)
	{
	struct vertexnode Q[10],u,v;
	int i,loc;
	for(i=0;i<n;i++)
		{
		V[i].key= mx;
		V[i].p = -1;
		}
	V[s].key= 0;
	QInsert(Q,V,n);
//	printQ(Q,qsize);
	printV(V,n);
	printQ(Q,qsize);
	while(qsize>0)
		{
		u = extract_min(Q);
		printf("\n\n %d extracted with key %d ",u.vertex,u.key);
		for(i=0;i<n;i++)
			{
			if((u.vertex!=i) && (w[u.vertex][i]>=1 && w[u.vertex][i]<mx))
				{
				v = V[i];
				loc=inQueue(Q,v);
				if((loc>=0 && loc< qsize) && w[u.vertex][i]<v.key )
					{
					printf(" Update =%d",i);
					V[i].key =	w[u.vertex][i];
					V[i].p= u.vertex;
					Q[loc].key=V[i].key;
					Q[loc].p=V[i].p;
					}
				}
			}
		printf("\n");
		sort(Q,qsize);
		printV(V,n);
		printQ(Q,qsize);
		}
	}

struct vertexnode extract_min(struct vertexnode Q[])
```

```
	{
	int i;
	struct vertexnode min;
	if(qsize<0)
		{
		printf("underflow");
		return Q[0];
		}
	else
		{
		min = Q[0];
		for(i=1;i<qsize;i++)
			Q[i-1]= Q[i];
		qsize--;
		return min;
		}
	}
void QInsert(struct vertexnode Q[],struct vertexnode V[],int n)
	{
	int i,j;
	struct vertexnode temp;
	qsize=n;
	for(i=0;i<n;i++)
		Q[i] = V[i];
	sort(Q,qsize);
	//printQ(Q,n);
	}
void sort(struct vertexnode Q[],int n)
	{
	int i,j;
		struct vertexnode temp;
	for(i=1;i<=qsize-1;i++)
		{
		for(j=0;j<qsize-i;j++)
			{
			if(Q[j].key>Q[j+1].key)
				{
				temp=   Q[j];
				Q[j] = Q[j+1];
				Q[j+1]= temp;
				}
			}
		}

	}
int inQueue(struct vertexnode Q[],struct vertexnode v)
	{
	int i;
```

```
for(i=0;i<qsize;i++)
        {
        if(v.vertex == Q[i].vertex)
                return i;
        }
return -1;
}
```