```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
int W[10][10], minlength = 987654321, n=5;
int  opttour[10], front = -1,rear=-1;


typedef struct node {
	int level=0;
	int bound=0;
	int path[10] = { 0, };
};
node PQ[100];
void init()
{
	node u;
	u.level = -1;
	u.bound = -1;
	PQ[0] = u;
}
int length(node v) {
	int length=0;
	int j = 0;
	while (v.path[j] != 0)
		j++;
	for (int i = 0;i<j; i++)
	{
		length += W[v.path[i]][v.path[i + 1]];
	}
	return length;
}
bool hasIncoming(int j, node v)
{
	int i = 0;
	while (v.path[i] != 0)
		i++;
	for (int k = 1; k < i ; k++)
	{
		if (v.path[k] == j)
			return 1;
	}
```

```
        return 0;
}
bool hasOutgoing(int i, node v)
{
        int j=0;
        while(v.path[j]!=0)
                j++;
                for (int k = 0; k < j-1; k++)
                {
                        if(v.path[k]==i)
                                return 1;
                }
        return 0;
}
int last(node v)
{
        int j = 0;
        while (v.path[j] != 0)
                j++;
        return v.path[j - 1];
}

int bound(node v){
        int min, total = length(v);
        for (int i = 1; i <= n; i++) {
                if (hasOutgoing(i, v))
                        continue;
                min = 987654321;
                for(int j = 1; j <= n; j++) {
                        if (i == j)
                                continue;
                        if (hasIncoming(j, v))
                                continue;
                        if (j == 1 && i == last(v))
                                continue;
                        if (min > W[i][j])
                                min = W[i][j];
                }
                total += min;
        }
        return total;
```

```
}
bool PQ_empty() {
	if (front == rear)
		return 1;
	else
		return 0;
}

void insert(node v)
{
	rear = rear + 1;
	PQ[rear] = v;
}
bool constains(node v, int i)
{
	int j=0;
	while (v.path[j] != 0) {
		if (v.path[j] == i)
			return 1;
		else
			j++;
	}
	return 0;
}
node ordered_set_insert(node u, int i) {
	int j = 0;
	while (u.path[j] != 0)
	{
		j++;
	}
	u.path[j] = i;
	return u;
}
node remain_set_insert(node u)
{
	int sum=0,i;
	for (i = 0; i <= n; i++)
	{
		sum += i;
	}
	i = 0;
```

```
        while (u.path[i] != 0)
        {
                sum -= u.path[i];
                i++;
        }
        u.path[i] = sum;
        return u;
}
node remove()
{
        front = front + 1;
        return PQ[front];
}
void tsp() {
        node u, v;
        int j=0, k=0;
        v.level = 0;
        v.path[0] = 1;
        v.bound = bound(v);
        insert(v);
        while (!PQ_empty()) {
                v = remove();
                if (v.bound < minlength)
                {
                        u.level = v.level + 1;
                        for (int i = 2; i <= n; i++) {
                                if (constains(v, i))
                                        continue;
                                for (int i = 0; i < 10; i++)
                                        u.path[i] = 0;
                                while (v.path[k] != 0)
                                        k++;
                                for(j=0;j<k;j++)
                                {
                                        u.path[j] = v.path[j];
                                }
                                u=ordered_set_insert(u, i);
                                if (u.level == n - 2) {
                                        u=remain_set_insert(u);
                                        u=ordered_set_insert(u, 1);
                                        if (length(u) < minlength) {
```

```
                                        minlength = length(u);
                                        j = 0;
                                        while (u.path[j] != 0)
                                        {
                                                opttour[j] = u.path[j];
                                                j++;
                                        }
                                }
                        }
                        else {
                                u.bound = bound(u);
                                if (u.bound < minlength)
                                        insert(u);
                        }
                }
        }
}
int main() {
        int i, j;

        printf("5*5 그래프의 인접 행렬식 가중치를 입력해주세요. \n");
        for (i = 1; i <= n; i++)
        {
                printf("%d행의 데이터 입력 \n", i);
                for (j = 1; j <= n; j++)
                        scanf("%d", &W[i][j]);
        }
        printf("\t인접행렬식 표현 : \n");
        for (i = 1; i <= n; i++)
        {
                printf("\n\n");
                for (j = 1; j <= n; j++)
                        printf("\t%d", W[i][j]);
        }
        tsp();
        printf("\n\n\t최적경로 : ");
        for (i = 0; i <= n; i++)
        {
                printf("%d  ", opttour[i]);
        }
```

```
        printf("\n\t최단비용값 : %d", minlength);
}
```