```
//  「C で学ぶデータ構造とアルゴリズム」(西原清一)  オーム社, 2008
//    図 5・32 (p.132) Floyd アルゴリズムで最短路を出力するプログラム
//    改訂版

#include<stdio.h>
#define     N   4
#define     S   1
#define     M   999   /*無限大*/
#define     L   100
#define     OVERFLOW    -1
#define     UNDERFLOW   -2

int w[N][N]={{0,9,4,M},{M,0,M,9},{M,M,0,2},{3,1,M,0}};
int p[N][N], d[N][N];
int stk[L],t;     /*stack*/

void Floyd()
{   int i,j,k,can;
    for (i=0; i<N; i++)
        for (j=0; j<N; j++) {d[i][j] = w[i][j]; p[i][j] = i;}
    for (k=0; k<N; k++)
        for (i=0; i<N; i++)
            for (j=0; j<N; j++) {
                can = d[i][k] + d[k][j];
                if (can < d[i][j]) {
                    d[i][j] = can; p[i][j] = p[k][j];}
            }
}

int push(int x)
{   if (t < L) stk[t++] = x;
    else return OVERFLOW;
    return 0;
}

int pop()
{   if (t > 0) return stk[--t];
    else return UNDERFLOW;
}
```

```
int empty()
{   if (t == 0) return 1; return 0;}

void shortest_path(int m, int n)  // 図5・32 (p.132) print the shortest path
{   int x;
    if (d[m][n] == M) printf("there is no path.¥n");
    else {x = n; push(x);
          while (x != m) {x = p[m][x]; push(x);}
          while (empty() != 1) {printf("%d =>",pop());}
          printf("END¥n");
    }
}

main()
{   int m,n;
    Floyd();
    t = 0;
    printf("shortest path from m to n. ¥nm? n?  ");
    scanf("%d%d",&m,&n);
    shortest_path(m,n);
}
```