```
#include <stdio.h>

#define N 5

#define M 1000

#define MAXSTACK 5


typedef int itemType;

//Definisi struktur stack

typedef struct {

     itemType item[MAXSTACK];      //Array yg berisi data
tumpukan

     int count;                                  //indeks data
paling atas dari stack

} Stack;


void copy_beban();

void beban_rute_jalur();

void jalur();

void tampil();

void baca_matriks();


void initializeStack(Stack *);

int empty(Stack *);

int full(Stack *);

void push(itemType, Stack *);

itemType pop(Stack *);


int Q[N][N] = { {M,1,3,M,M},

                     {M,M,1,M,5},
```

```
                    {3,M,M,2,M},
                    {M,M,M,M,1},
                    {M,M,M,M,M}};

int P[N][N]= { {0,1,1,0,0},
                    {0,0,1,0,1},
                    {1,0,0,1,0},
                    {0,0,0,0,1},
                    {0,0,0,0,0}};

int R[N][N]= { {M,0,0,M,M},        //krn indeks array dimulai
dari 0
                    {M,M,0,M,0},        //pdhl nilai k : 1 to
n
                    {0,M,M,0,M},
                    {M,M,M,M,0},
                    {M,M,M,M,M}};
int i, j, k;

main()
{
     char jawab;

     printf("MATRIKS AWAL\n");
     tampil();
     beban_rute_jalur();
     //jalur();
     printf("\nMATRIKS SETELAH PROSES\n");
```

```
     tampil();
     puts("");
     puts("MEMBACA MATRIKS RUTE");
     do {
          baca_matriks();
          puts("");
          fflush(stdin);
          printf("Mau baca lagi (y/t) ? ");
          scanf("%c", &jawab);
     } while (jawab == 'y' || jawab == 'Y');
}


void beban_rute_jalur()
{
     for(k=0; k<N; k++)
          for(i=0; i<N; i++)
               for(j=0; j<N; j++)
               {
                    if ((Q[i][k]+ Q[k][j]) < Q[i][j]){
                         P[i][j]= P[i][j] || (P[i][k] &&
P[k][j]);
                         Q[i][j] = Q[i][k]+Q[k][j];
                         if (R[k][j] == 0)
                              R[i][j] = k+1; //krn indeks k
dimulai dari 0
                         else
                              R[i][j]=R[k][j];
```

```
                    }
                }
}

void tampil()
{
     printf("Matriks beban:\n");
     for (i=0; i<N; i++) {
          for (j=0; j<N; j++) {
               if (Q[i][j] == M)
                    printf("M ");
               else
                    printf("%d ", Q[i][j]);
          }
          printf("\n");
     }

     printf("\nMatriks jalur\n");
     for (i=0; i<N; i++) {
          for (j=0; j<N; j++) {
               printf("%d ",P[i][j]);
          }
          printf("\n");
     }

     printf("\nMatriks rute\n");
     for (i=0; i<N; i++) {
```

```
            for (j=0; j<N; j++) {

                if (R[i][j] == M)
                    printf("M ");
                else
                    printf("%d ",R[i][j]);
            }
            printf("\n");
        }
}

void baca_matriks()
{
        int asal, tuj, tmp, x, tuj2;
        Stack tumpukan;

        initializeStack(&tumpukan);
        printf("Masukkan node asal : ");
        scanf("%d", &asal);
        fflush(stdin);
        printf("Masukkan node tujuan : ");
        scanf("%d", &tuj);
        puts("");
        tuj2= tuj;

        do {
            tmp = R[asal-1][tuj-1];
```

```
            if (tmp == M)
                  break;
            if(tmp != 0){
                  push(tmp, &tumpukan);
                  tuj = tmp;
            }
      } while(tmp != 0);        //0 = stop

      if(tmp != M) {
            printf("Rute dari %d ke %d adalah : ", asal, tuj2);
            printf("%d - ", asal);
            while(!empty(&tumpukan))
            {
                  x = pop(&tumpukan);
                  printf("%d - ", x);
            }
            printf("%d", tuj2);
      }
      else
            printf("Tidak ada rute dari node %d ke %d\n", asal,
tuj2);
}

void initializeStack(Stack *S)
{
      S->count = 0;
}
```

```
int empty(Stack *S)
{
     return (S->count == 0);
}


int full(Stack *S)
{
     return (S->count == MAXSTACK);
}

void push(itemType x, Stack *S)
{
     if (full(S)) {      //stack penuh
          puts("\nSTOP!!...");
          puts("Stack PENUH!! Data terakhir gak bisa masuk");
     }
     else {
          ++(S->count);
          S->item[S->count] = x;
  }
}

itemType pop(Stack *S)
{
     char x;
```

```
    if (empty(S)) {          //stack kosong
    puts ("Stack masih kosong!");
         return 0;
    }
    else  {
         x = (S->item[S->count]);
         --(S->count);
         return x;
  }
}
```