```
//========================================== create fluxes at faces
between CV's
#include "global.h"
#include <math.h>
#include <stdio.h>

double conduct(double p);  // declare function

void flux(const int W, double Fx[][W+1], double Fy[][W+1], double T[]
[W+2], double p[][W+1], double F0[]){
        double k, ki, kim, R;
        // Boundaries
                // Left and right
        for(int j = 1; j <= M; j++){
                k = conduct(p[1][j]);
                R = dx/(2*k);
                //Fx[0][j] = -(T[1][j] - Q0)/R; //F_1/2
       Fx[0][j] = - (T[0][j] - Tinf)/(R + 1/h); // left; convective
2-D
                k = conduct(p[M][j]);
                R = dx/(2*k);
                //F[M] = - (T[M] - Tinf)/(R + 1/h); // convective 1-D
                //Fx[M][j] = - (T[M+1][j] - T[M][j])/R; // insulated
2-D
                Fx[M][j] = (T[M][j] - Tinf)/(R + 1/h); // right;
convective 2-D
        }
                // Bottom and top
        for(int i = 1; i <= M; i++){
                k = conduct(p[i][1]);
                R = dx/(2*k);
                //Fy[i][0] = -(T[i][1] - Q0)/R; //F_1/2
       Fy[i][0] = - (T[i][0] - Tinf)/(R + 1/h); // bottom; convective
2-D
                //==================== TOP
=======================================
                k = conduct(p[i][M]);
                R = dx/(2*k);
                if(!BCType){ // checks boundary condition type
                        Fy[i][M] = (T[i][M] - F0[i])/R; // const temp
BC
                        //printf("TEMPBC, Flux = %f\n", Fy[i][M]);
                }else{
                        Fy[i][M] = - F0[i]/(dx);  // const Flux BC
                        //printf("FLUXBC, Flux = %f\n", Fy[i][M]);
                }
                //Fy[i][M] = (T[i][M] - Tinf)/(R + 1/h); //
convective 2-D
        }
```

```
        // LEFT TO RIGHT FLUX
        for(int i = 2; i <= M; i++){ // 2-1/2 < i-1/2 < M-1/2
                for(int j = 1; j <= M; j++){
                        // get k and R
                        ki = conduct(p[i][j]); // ki
                        kim = conduct(p[i-1][j]); // ki-1
                        R = dx/(2*ki) + dx/(2*kim); // Ri-1/2
                        // compute flux
                        Fx[i-1][j] = -(T[i][j] - T[i-1][j])/R; //
F_i-1/2
                }
        }
        // UP TO DOWN FLUX
        for(int i = 1; i <= M; i++){ // 2-1/2 < i-1/2 < M-1/2
                for(int j = 2; j <= M; j++){
                        // get k and R
                        ki = conduct(p[i][j]); // ki
                        kim = conduct(p[i][j-1]); // ki-1
                        R = dx/(2*ki) + dx/(2*kim); // Ri-1/2
                        // compute flux
                        Fy[i][j-1] = -(T[i][j] - T[i][j-1])/R; //
F_i-1/2
                }
        }
}
```