```
//======================== print solution profile to text file at
certain times
#include "global.h"
#include <stdio.h>
void output(const int W, double X[], double Y[], double T[][W+2],
double Fx[][W+1], double Fy[][W+1], double E[][W+1], double p[][W+1],
double time, int nsteps, double tend){
//      double flux, energy, phase;
        FILE *OUT;
        FILE *TEMP;
        FILE *PHASE;
        FILE *ENTH; // initialize file var
        if(!nsteps){ // if haven't begun time stepping
                OUT = fopen("outputs/values.o", "w"); // new file
                fprintf(OUT,"#nstep time (ms)  width (cm) depth(cm)
energy (J)\n");
                TEMP = fopen("outputs/temp.o", "w"); // new file
                //fprintf(TEMP,"#Temperatures by timesteps\n");
                PHASE = fopen("outputs/phase.o", "w"); // new file
                //fprintf(PHASE,"#Liquid fraction by timestep\n");
                ENTH = fopen("outputs/enth.o", "w"); // new file
                //fprintf(ENTH,"#Enthalpies by timestep\n");
        }
        else{
                OUT = fopen("outputs/values.o", "a"); // old file
                TEMP = fopen("outputs/temp.o", "w"); // old file
                PHASE = fopen("outputs/phase.o", "w"); // old file
                ENTH = fopen("outputs/enth.o", "w"); // old file
        }
        // hidden time, steps, error
        // position and profile in columns
        //================================================
TEMPERATURES
        // inside temperatures only, walls not necessary
        for(int j = 1; j <= M; j++){
                for(int i = 1; i <= M; i++){
                        fprintf(TEMP, "%22.15e ", T[i][j]);
                }
                fprintf(TEMP, "\n");
        }
        fprintf(TEMP, "\n");
        fclose(TEMP);
        //================================================ PHASES
                for(int j = 1; j <= M; j++){
                        for(int i = 1; i <= M; i++){
                                fprintf(PHASE, "%22.15e ", p[i][j]);
                        }
                fprintf(PHASE, "\n");
        }
        fprintf(PHASE, "\n");
```

```
        fclose(PHASE);
        //================================================ ENTHALPIES
        // just for funsies
        for(int j = 1; j <= M; j++){
                for(int i = 1; i <= M; i++){
                        fprintf(ENTH, "%22.15e ", E[i][j]);
                }
                fprintf(ENTH, "\n");
        }
        fprintf(ENTH, "\n");
        fclose(ENTH);
        //================================================
TIMESTAMPS/ETC
        double xl, xr, yd, width, depth, energy;
        int il = M/2, ir = M/2, id = M;
        for(int l = M/2; l <= M; l++){
                for(int k = 1; k <= M; k++){ // bounds for width of
pool
                        if(p[k-1][l] < 1 && p[k+1][l] == 1){
                                if(k < il){
                                        il = k;}
                        }
                        if(p[k-1][l] == 1 && p[k+1][l] < 1){
                                if(k > ir){
                                        ir = k;}
                        }
        }
        }
        for(int k = 1; k <= M; k++){
                if(p[M/2][k-1] < 1 && p[M/2][k+1] == 1){
                        id = k;}
        }
        xl = a + ((double)il - 0.5)*dx;
        xr = a + ((double)ir - 0.5)*dx;
        yd = a + ((double)id - 0.5)*dx;
        width = 100*(xr - xl); // in cm
        depth = 100*(b - yd); // in cm
        energy = Q0*time; // in Joules
        fprintf(OUT, "%i %6.4f %6.4f %6.4f %6.4f\n", nsteps,
1000*time, width, depth, energy);
        //fprintf(OUT, "# Error up to this time: %.15e\n\n", ERR);
        //
        // COMPLETION BAR
        if(nsteps == 0){
                printf("0%%       20%%       40%%       60%%
80%%        100%%\n");
                printf(" ");
        }
        if( time > nbar*tend/50 ){
                //printf("%3.0f%%\n", 100*time/tend);
```

```
                printf("=");
                fflush(stdout);
                nbar = nbar + 1;
        }
        if(width > 2){
                maxwidth = 1;
                int barleft = 50 - nbar;
                for(int g = 0; g <= barleft; g++){
                printf("=");}
        }


        /*for(int j = M+1; j >= 0; j--){
                for(int i = 0; i <= M+1; i++){
                        fprintf(OUT, "%22.15e ", T[i][j]);
                }
                fprintf(OUT, "\n");
        }*/


        /*for(int i = 0; i <= M+1; i++){
                if(i == 0){
                        fprintf(OUT, "%18.15e %18.15e %18.15e\n",
X[i], T[i], F[i]);
                }
                if(i > 0 && i <= M){
                        fprintf(OUT, "%18.15e %18.15e %18.15e %18.15e
%.3f\n", X[i], T[i], F[i], E[i], p[i]);
                }else if(i > M){
                        fprintf(OUT, "%.15e %.15e\n", X[i], T[i]);
                }
        }*/
        //fprintf(OUT, "\n"); // new line to separate times of
printing
        fclose(OUT);
}
```