#include<iostream>

float uae[1000][1000]; // for ee

float uan[1000][1000]; //for nne

float uas[1000][1000]; //for sse

float uaw[1000][1000]; //for w

float uap[1000][1000]; //for e

float vae[1000][1000];

float van[1000][1000];

float vas[1000][1000];

float vaw[1000][1000];

float vap[1000][1000];

float bu[1000][1000];

float bv[1000][1000]; //coefficients for u and v equations

float u[1000[1000];

float v[1000][1000];

float uh[1000][1000];

float vh[1000][1000]; //u hat and v hat

float p[1000][1000]; //pressure

float bp[1000][1000];

float pae[1000][1000];

float pan[1000][1000];

float pas[1000][1000];

float paw[1000][1000];

float pap[1000][1000]; //coefficients for pressure equation

float rho, vis, g; //density, viscocity, gravity

float delx; //cell size

float n; //grid size in any directon. Assuming square grid

//NOTE : CONVENTION : INCREASING i IS +VE X DIRECTION AND INCREASING J IS +VE Y DIRECTION. IE.,CODE STARTS AT SOUTH-WEST CORNER OF GRID.

void gaussseidel(float ap[1000][1000],float ae[1000][1000],float aw[1000][1000],float an[1000][1000],float as[1000][1000],float b[1000][1000], float x[1000][1000]);

int main()

{

// SIMPLER ALGORITHM FOR BACK STEP FLOW

int i,j;

for(i=1;i<= n;i++)

{

for(j=1;j<= n;j++)

{

uh[i][j] = (uae[i][j]\*u[i+1][j] + uaw[i][j]\*u[i-1][j] + uas[i][j]\*u[i][j-1] + uan[i][j]\*u[i][j+1])/uap[i][j] ;

vh[i][j] = (vae[i][j]\*v[i+1][j] + vaw[i][j]\*vaw[i-1][j] + vas[i][j]\*v[i][j-1] + van[i][j]\*v[i][j+1])/vap[i][j] ; //STEP 2 OF SIMPLER

}

}

for(i=1;i<= n ;i++)

{

for(j=1;j<= n ;j++)

{

pae[i][j] = rho\*(delx^2)/(4\*vis);

pan[i][j] = pas[i][j] = paw[i][j] = pae[i][j];

pap[i][j] = 4\*pae[i][j];

bp[i][j] = delx\*rho\*(uh[i][j]-uh[i+1][j] + vh[i,j] - vh[i,j+1]) ; // The west face is i, the east face is i+1, north face is j+1, south face is j

} //STEP 3 OF SIMPLER

}

gaussseidel(pap,pae,paw,pan,pas,bp,p); //STEP 3 OF SIMPLER

for(i=1;i<=n; i++)

{

for(j= 1; j<=n;j++)

{ //initialise coefficients for x momentum equation

if(u[i+1][j] + u[i][j] >0) //UPWINDING IN U

uae[i][j] = vis;

else

uae[i][j] = vis - rho\*delx\*u[i+1][j];

if(u[i-1][j] + u[i][j] >0)

uaw[i][j] = vis - rho\*delx\*u[i-1][j];

else

uaw[i][j] = vis;

if(v[i][j] + v[i+1][j] > 0) //UPWINDING IN V

uan[i][j] = vis ;

else

uan[i][j] = vis - rho\*delx\*(v[i][j] + v[i+1][j] + v[i][j+1] + v[i+1][j+1])/4 ;

if(v[i][j-1] + v[i+1][j-1] > 0)

uas[i][j] = vis - rho\*delx\*(v[i][j-1] + v[i+1][j-1] + v[i,j-2] + v[i+1,j-2])/4;

else

uas[i][j] = vis;

bu = (p[i][j] - p[i+1][j])\*delx;

vap[i][j] = 4\*vis + rho\*delx\*u[i+1][j] - rho\*delx\*u[i-1][j] + rho\*delx\*(v[i][j] + v[i+1][j] + v[i][j+1] + v[i+1][j+1])/4 - rho\*delx\*(v[i][j-1] + v[i+1][j-1] + v[i,j-2] + v[i+1,j-2])/4 ;

//initialise constants for y momentum equation

if(v[i][j+1] + v[i][j] >0)

van[i][j] = vis;

else

van[i][j] = vis - rho\*delx\*v[i][j+1]; //UPWINDING IN V

if(v[i][j-1] + v[i][j] >0)

vas[i][j] = vis - rho\*delx\*v[i][j-1];

else

vas[i][j] = vis;

if(u[i][j+1] + u[i][j] > 0)

vae[i][j] = vis;

else

vae[i][j] = vis - rho\*delx\*(u[i][j] + u[i][j+1] + u[i+1][j] + u[i+1][j+1])/4;

if(u[i-1][j] + u[i-1][j+1] > 0)

vaw[i][j] = vis - rho\*delx\*(u[i-2][j] + u[i-2][j+1] + u[i-1][j] + u[i-1][j+1])/4;

else

vaw[i][j] = vis;

bv = (p[i][j] - p[i][j+1])\*delx - rho\*g;

vap[i][j] = 4\*vis + rho\*delx\*v[i][j+1] - rho\*delx\*v[i][j-1] + rho\*delx\*(u[i][j] + u[i][j+1] + u[i+1][j] + u[i+1][j+1])/4 - rho\*delx\*(u[i-2][j] + u[i-2][j+1] + u[i-1][j] + u[i-1][j+1])/4 ;

}

}

gaussseidel(uap,uae,uaw,uan,uas,bu,u);

gaussseidel(vap,vae,vaw,van,vas,bv,v);

}