Une possible solution de la question 1.3. f) :

```
// la frontière du domaine
border L1(t = 0, 1){x=2*t; y=0; label=1;}
border L2(t = 0, 1){x=2; y=t; label=1;}
border L3(t = 1, 0){x=2*t; y=1; label=1;}
border L4(t = 1, 0){x=0; y=t; label=1;}

// les fonctions intervenant dans le problème
func c = 1;
func f = (5*pi*pi + 1)*sin(pi*x)*sin(2*pi*y);
func uex = sin(pi*x)*sin(2*pi*y);

// un maillage
mesh Th = buildmesh( L1(20) + L2(10) + L3(20)+L4(10) );

// l'espace P1
fespace Vh(Th, P1);
Vh u, v;

// l'espace P2
fespace Vh2(Th, P2);
Vh2 u2, w;

// le problème variationnelle P1
problem monPb(u, v) = int2d(Th)(dx(u)*dx(v) + dy(u)*dy(v) + c*u*v)
          - int2d(Th)(f*v) + on(1, u=0);

// le problème variationnelle P2
problem monPb2(u2, w) = int2d(Th)(dx(u2)*dx(w) + dy(u2)*dy(w) + c*u2*w)
             - int2d(Th)(f*w) + on(1, u2=0);

int[int] p(5); // le vecteur contenant le nombre de points
real[int] errP1L2(5), errP1H1(5), errP2L2(5), errP2H1(5);
for(int k=0; k < 5; k++){
  p[k] = 10*2^k; // nb de points
  cout << "p[k] = " << p[k] << endl; //affichage
  // nouveau mesh
  Th = buildmesh( L1(2*p[k]) + L2(p[k]) + L3(2*p[k])+L4(p[k]) );
  monPb;              // problème P1
  Vh err = uex - u;
  errP1H1[k] = sqrt(int2d(Th)(dx(err)*dx(err) + dy(err)*dy(err)));
  errP1L2[k] = sqrt(int2d(Th)(err*err));
  monPb2;             // problème P2
  Vh2 err2 = uex - u2;
  errP2H1[k] = sqrt(int2d(Th)(dx(err2)*dx(err2) + dy(err2)*dy(err2)));
  errP2L2[k] = sqrt(int2d(Th)(err2*err2));
}

// affichage des erreurs dans la console
for (int k=0; k < 5; k++){
  cout << "Erreur P1H1 : " << errP1H1[k] << endl;
  cout << "Erreur P1L2 : " << errP1L2[k] << endl;
  cout << "Erreur P2H1 : " << errP2H1[k] << endl;
  cout << "Erreur P2L2 : " << errP2L2[k] << endl;
}
```

Une possible solution pour la partie 2 :

```
// la frontière Gamma0 -- condition Dirichlet homogène
border Gamma01(t=-2,2){x=t; y=-2; label=1;}
border Gamma02(t=-2,2){x=2; y=t; label=1;}
border Gamma03(t=2,-2){x=t; y=2; label=1;}
border Gamma04(t=2,-2){x=-2; y=t; label=1;}

// la frontière Gamma1 -- condition Neumann non-homogène
border Gamma1(t=2*pi,0){x=cos(t); y=sin(t); label=2;}

// un maillage
mesh Th = buildmesh(Gamma01(40) + Gamma02(40) + Gamma03(40) +Gamma04(40) + Gamma1(80));
// plot(Th, wait=true); // on peut afficher le maillage

// les fonctions f, c, g
func f=sin(pi*x)*sin(pi*y);
func c=x*x + y*y;
func g=x^2 + y^2;

// l'espace d'éléments finis de type P1
fespace Vh1(Th, P1);
// l'espace d'éléments finis de type P1
fespace Vh2(Th, P2);

// Nous pouvons afficher la fonction projétée sur Vh2
// Vh2 gh = g;
// plot(gh, nbiso=50, wait=true);

// la formulation varitionnelle
Vh2 u, v;
problem monPb(u, v) = int2d(Th)(dx(u)*dx(v) + c*dy(u)*dy(v))
                      -int2d(Th)(f*v) -int1d(Th, 2)(g*v)+ on(1, u=0);
monPb;

// on affiche la solution
plot(u, nbiso=75, wait=true);
```

Possibles développements pour les comptes rendus :

– donner une solution détaillée pour la question 1.1.
– reprendre la même démarche de la question 1.3. f) pour un autre choix de fonctions f et c. Expliquer les résultats obtenus.
– écrire le problème (3) sous forme variationnelle (question 2.1). Justifier l'existence et l'unicité des solutions.
– la question 2.2 e).