Contenu de main.h :

```
#include <iostream.h>
void make_generation (TabTD source, TabTD dest, int r, int s);
```

Contenu de main.cpp :

```
#include "tabmanip.h"
#include "main.h"

void main ()
{
  // Exemple de tableau de depart
  TabTD tableau1 = {
    {
     {0, 1, 0, 1, 1, 0, 1},
     {0, 1, 0, 0, 1, 1, 0},
     {1, 0, 0, 1, 0, 1, 0},
     {0, 1, 0, 1, 0, 1, 0},
     {0, 0, 1, 0, 0, 1, 0},
     {0, 1, 0, 1, 0, 1, 0}
     },
    {
     {0, 0, 1, 0, 0, 1, 0},
     {1, 0, 1, 0, 1, 0, 1},
     {0, 0, 0, 1, 0, 1, 0},
     {0, 1, 1, 0, 1, 0, 1},
     {1, 0, 1, 0, 1, 0, 1},
     {1, 1, 0, 1, 0, 1, 1}
     },
    {
     {0, 1, 0, 1, 0, 1, 1},
     {1, 1, 0, 1, 0, 1, 1},
     {1, 1, 0, 1, 1, 1, 0},
     {1, 1, 0, 1, 1, 1, 0},
     {1, 0, 1, 1, 1, 0, 1},
     {1, 0, 1, 1, 1, 0, 0}

     },
    {
     {0, 1, 0, 1, 1, 1, 1},
     {0, 1, 0, 1, 1, 0, 1},
     {0, 1, 0, 0, 0, 1, 0},
     {1, 1, 1, 1, 0, 0, 0},
     {1, 0, 1, 1, 1, 1, 1},
     {1, 1, 1, 1, 1, 1, 1}
     }
  };
  /* Initialise un second tableau afin de pouvoir stocker les resultats lors de la
     recherche de la generation suivante. */
  TabTD tableau2;

  /* On cree ici deux pointeurs qui poitent respectivement vers chacun des tableaux afin
     de pouvoir les echanger par la suite (On evite ainsi de recopier le contenu du
     deuxieme tableau vers le premier en les echangeant simplement). */
  TabTD *ptab1 = &tableau1;
  TabTD *ptab2 = &tableau2;
  int choix, r, s, num_gene;
  char etat;
  // On lis au clavier les donnees necessaires au probleme.
  cout << "Entrez 1 pour utiliser l'exemple, entrez n'importe quelle " <<
    "autre valeur pour definir la configuration initiale" << endl;
  cin >> choix;
  if (choix != 1)
    {
```

```
      cout << "Entrez la configuration de depart du tableau tridimensionel. " <<
        "(1 signifie qu'un organisme est present dans la case, 0 signifie qu'aucun n'est present)"
        << endl;
      lecture_tableau (*ptab1);
    }
  cout << "Pour qu'un organisme puisse survivre, le nombre d'organismes adjacents " <<
    "doit etre compris entre 2 et une valeur donnee; Entrez cette valeur." << endl;
  cin >> r;
  cout << "Pour naitre un organisme a besoin d'exactement un nombre " <<
    "donne d'organismes adjacents, entrez ce nombre." << endl;
  cin >> s;
  cout << "Entrez le nombre de generations a verifier" << endl;
  cin >> num_gene;

  // On affiche l'etat initial du tableau.
  cout << "Vue initiale:" << endl;
  affichage_tableau (*ptab1);

  /* On recherche autant de fois que demande la generation suivante */
  for (int i = 0; i < num_gene && etat != 's'; ++i)
    {
      TabTD *Save;
      make_generation (*ptab1, *ptab2, r, s);
      /* Plutot que de recopier le resultat dans le tableau initial, on echange
         simplement les pointeurs afin de considerer le tableau resultat comme etant le
         tableau initial. */
      Save = ptab1;
      ptab1 = ptab2;
      ptab2 = Save;
      cout << "Generation " << (i + 1) << ":" << endl;
      affichage_tableau (*ptab1);        // Affiche la disposition des organisme pour
                                         // chaque generation
      cout << "Entrez s pour arreter ou n'importequoi d'autre pour continuer." << endl;
      cin >> etat;
    }
}

void make_generation (TabTD source, TabTD dest, int r, int s)
{
  /* Fonction recherchant pour chaque case le nombre d'organismes adjacents. Si un
     organisme est present dans la case, on évrifie que le nombre d'organismes adjacents
     est compris entre 2 et r, si ce n'est pas le cas, l'organisme meurt. Si aucun
     organisme n'est present, on le cree si le nombre d'organismes adjacents est
     exactement egal a s. */
  for (int i = 0; i < MAXX; ++i)
    for (int j = 0; j < MAXY; ++j)
      for (int k = 0; k < MAXZ; ++k)
        {
          int numadj;
          numadj = occ (source, i, j, k);        // Fait appel a la fonction occ pour
                                                 // compter les organismes adjacents.
          if (source[i][j][k])
            {
              dest[i][j][k] = (numadj >= 2 && numadj <= r);
            }
          else
            {
              dest[i][j][k] = (numadj == s);
            }
        }

}
```

Contenu de tabmanip.h :

```
#include <iostream.h>
const int MAXX = 4;
const int MAXY = 6;
const int MAXZ = 7;
typedef bool TabTD[MAXX][MAXY][MAXZ];
void affichage_tableau (TabTD matrice);
int occ (TabTD tableau, int pos_x, int pos_y, int pos_z);
void lecture_tableau (TabTD matrice);
```

Contenu de tabmanip.cpp :

```
#include "tabmanip.h"
#include "iostream.h"
void lecture_tableau (TabTD tableau)
{
  /* Fonction lisant au clavier la configuration initiale, demandant l'etat de chaque
     case */
  for (int i = 0; i < MAXX; ++i)
    {
      cout << "Couche:" << i << " ";
      for (int j = 0; j < MAXY; ++j)
        {
          cout << "Ligne:" << j << endl;
          for (int k = 0; k < MAXZ; ++k)
            cin >> tableau[i][j][k];
        }
    }
}

void affichage_tableau (TabTD tableau)
{
  /* Fonction affichant a l'ecran l'etat du tableau 3D */
  for (int i = 0; i < MAXX; ++i)
    {
      for (int j = 0; j < MAXY; ++j)
        {
          for (int k = 0; k < MAXZ; ++k)
            cout << tableau[i][j][k] << " ";
          cout << endl;
        }
      cout << endl;
    }
}

int occ (TabTD tableau, int pos_x, int pos_y, int pos_z)
{
  /* Cette fonction compte le nombre d'organismes presents dans les cases adjacentes */
  int inf_x, inf_y, inf_z, sup_x, sup_y, sup_z, sum = 0;
  // Pour ce faire, on trouve l'intervale a tester en verifiant si la case centrale se
  // trouve sur une face ou une arrete.
  if (pos_x > 0)
    inf_x = pos_x - 1;
  else
    inf_x = pos_x;
  if (pos_y > 0)
    inf_y = pos_y - 1;
  else
    inf_y = pos_y;
  if (pos_z > 0)
    inf_z = pos_z - 1;
  else
    inf_z = pos_z;
  if (pos_x < MAXX)
```

```
    sup_x = pos_x + 1;
  else
    sup_x = pos_x;
  if (pos_y < MAXY)
    sup_y = pos_y + 1;
  else
    sup_y = pos_y;
  if (pos_z < MAXZ)
    sup_z = pos_z + 1;
  else
    sup_z = pos_z;
// Boucles testant toutes les cases adjacentes
  for (int i = inf_x; i <= sup_x; ++i)
    for (int j = inf_y; j <= sup_y; ++j)
      for (int k = inf_z; k <= sup_z; ++k)
        {
          // Ajoute 1 si la case contient un organisme ET n'est pas la case centrale.
          if (tableau[i][j][k] && (i != pos_x || j != pos_y || k != pos_z))
            sum++;
        }

  return sum;
}
```