

Corrigé de l'examen, Programmation I
Sciences et Technologies du Vivant, Semestre 1
Chimie et Génie Chimique, Semestre 3

Exercice 1 - Sudoku

```
(a) void affiche(int grille[10][10]) {
    for (int i=1; i<10; i++) {
        for (int j=1; j<10; j++)
            if (grille[i][j] == 0)
                cout << ".";
            else
                cout << grille[i][j];
        cout << endl;
    }
}

(b) void supprime_possibilites(int grille[10][10], bool poss[10][10][10]) {
    int i0, j0;

    for (int i=1; i<10; i++)
        for (int j=1; j<10; j++)
            for (int k=1; k<10; k++)
                poss[i][j][k] = true;

    for (int i=1; i<10; i++)
        for (int j=1; j<10; j++)
            if (grille[i][j] > 0) {
                // mettre toutes les possibilites de la case a false (puisqu'elle
                // est deja determinee)
                for (int k=1; k<10; k++)
                    poss[i][j][k] = false;

                // ligne
                for (int k=1; k<10; k++)
                    poss[i][k][grille[i][j]] = false;

                // colonne
                for (int k=1; k<10; k++)
                    poss[k][j][grille[i][j]] = false;

                // carre
                i0 = i - ((i-1) % 3);
```

```

        j0 = j - ((j-1) % 3);
        for (int k=0; k<3; k++)
            for (int l=0; l<3; l++)
                poss[i0+k][j0+l][grille[i][j]] = false;
    }
}

```

```

(c) bool cherche_coup(bool poss[10][10][10], int * lig, int * col, int * elt) {
    int possibilites;
    int element;

    for (int i=1; i<10; i++)
        for (int j=1; j<10; j++) {
            possibilites = 0;
            for (int k=1; k<10; k++)
                if (poss[i][j][k]) {
                    possibilites++;
                    element = k;
                }
            if (possibilites == 1) {
                *lig = i;
                *col = j;
                *elt = element;
                return true;
            }
        }
    return false;
}

```

Exercice 2 - Mutations

```

(a) for (int i=0; i<p.nb_acides; i++)
    cout << p.acides[i].nom << endl;

(b) char *proteine_vers_adn(Proteine *p) {
    char *adn = new char[p->nb_acides * 3 + 1];
    for (int i=0; i<p->nb_acides; i++)
        strncpy(adn + 3*i, p->acides[i].codon, 3);
    adn[p->nb_acides * 3] = '\0';

    return adn;
}

(c) Acide_amine *cherche_acide(Acide_amine *acides, int nb_total_acides,
    char codon1, char codon2, char codon3) {
    for (int i=0; i<nb_total_acides; i++)
        if (acides[i].codon[0] == codon1 && acides[i].codon[1] == codon2
            && acides[i].codon[2] == codon3)

```

```

        return acides + i;

    return NULL;
}

```

```

(d) Proteine * adn_vers_proteine(char * ADN, Acide_amine * acides,
                                int nb_total_acides) {
    Proteine *p = new Proteine;
    p->nb_acides = strlen(ADN) / 3;
    p->acides = new Acide_amine[p->nb_acides];

    Acide_amine *a;

    for (int i=0; i<p->nb_acides; i++) {
        a = cherche_acide(acides, nb_total_acides, ADN[3*i], ADN[3*i+1], ADN[3*i+2]);
        strcpy(p->acides[i].nom, a->nom);
        strncpy(p->acides[i].codon, a->codon, 3);
    }

    return p;
}

```

```

(e) char *mutation(char *ADN, int nb_mutations) {
    int taille_adn = strlen(ADN);
    int nucleotide_a_muter, nouveau_nucleotide;
    char *nucleotides = "ACGT";
    char *adn_mutee = new char[taille_adn+1];
    strcpy(adn_mutee, ADN);

    for (int m=0; m<nb_mutations; m++) {
        nucleotide_a_muter = rand() % taille_adn;

        nouveau_nucleotide = rand() % 4;
        adn_mutee[nucleotide_a_muter] = nucleotides[nouveau_nucleotide];
    }
    return adn_mutee;
}

```

Exercice 3 - Pointeurs

- (a) a1) 1 1 1
 a2) 2 1 2
 a3) 2 2 2
 Segmentation fault
- (b) b1) 1 5
 b2) 1 1
 b3) 1 6

b4) 6 5
b5) 0 5

(c) c1) 0
c2) 1
c3) 3
c4) 4
c5) 3

Exercice 4 - Courbes de Lissajou

```
void affiche_lissajou(SimpleWindow *w, int A, float a, float b, float o,
    int x0, int y0) {
    float pas = 2 * M_PI / nbr_pas;
    float t = 0;

    // premier point
    int x1 = x0 + int(A * sin(o));
    int y1 = y0 - int(A * sin(o));
    int x2, y2;

    for (int i=1; i<nbr_pas; i++) {
        t += pas;
        x2 = x0 + int(A * sin(a * t + o));
        y2 = y0 - int(A * sin(b * t));
        w->draw_line(x1, y1, x2, y2);

        x1 = x2;
        y1 = y2;
    }
}
```