Correction des exercices sur les types structurés

1.
```
typedef struct {
    char nom[32];
    char prenom[32];
    float salaire;
} employe;

void promotion(float augmentation, employe *emp) {
    emp->salaire += augmentation;
}

int main(){
    employe entreprise[3];
    strcpy(entreprise[0].nom, "Doe");
    strcpy(entreprise[0].prenom, "John");
    entreprise[0].salaire = 1900;
    strcpy(entreprise[1].nom, "Renaud");
    strcpy(entreprise[1].nom, "Alice");
    entreprise[1].salaire = 2200;
    strcpy(entreprise[2].nom, "Herbert");
    strcpy(entreprise[2].nom, "Bob");
    entreprise[2].salaire = 2300;

    promotion(&entreprise[0]);
}
```

2.
```
typedef struct {
    double x;
    double y;
} point;

typedef struct {
    point origine;
    point destination;
} segment;

segment *creer_segment(double x1, double y1,
                       double x2, double y2) {
    segment *p;

    p = malloc(sizeof(segment));
    if (!p)
        return NULL;

    p->origine.x = x1;
    p->origine.y = y1;
    p->destination.x = x2;
    p->destination.y = y2;

    return p;
}

void liberer_segment(segment *p) {
    free(p);
}
```

3.
```
struct cellule_t {
    int value;
    struct cellule_t *suivant;
};
typedef struct cellule_t cellule;

void nouveau(cellule *premier, int n_valeur) {
    while (premier->suivant)
        premier = premier->suivant;
    cellule *n_cellule;
    n_cellule = malloc(sizeof(cellule));
    if (!n_cellule)
        return;
    n_cellule->next = NULL;
    n_cellule->valeur = n_valeur;
    premier->next = n_cellule;
}

void liberer_cellule(cellule *cell) {
    if (cell->next)
        liberer_cellule(cell->next);
    free(cell);
}
```

4.
```
typedef struct {
    unsigned nb_mots;
    char **mots;
} sequence_mots;

sequence_mots *decomposer_chaine(char *c) {
    sequence_mots *s;
    char *p = c;
    unsigned nb_char = 0;

    s = malloc(sizeof(sequence_mots));
    if (!s)
        return NULL;
    s->nb_mots = 0;

    while (*p)
        if (*p == ' ')
            p++;
        else {
            s->nb_mots++;
            for (; *p && *p != ' '; nb_char++, p++);
        }

    if (s->nb_mots == 0)
        return s;

    s->mots = malloc(s->nb_mots*sizeof(char*));
    if(!s->mots) {
        free(s);
        return NULL;
    }

    p = malloc(nb_char + s->nb_mots);
    if (!p) {
        free(s->nb_mots);
        free(s);
        return NULL;
    }

    for (unsigned i = 0; *c;)
        if (*c == ' ')
            c++;
        else {
            for (s->mots[i++] = p; *c && *c != ' '; c++)
                *p++ = *c;
```

```
            p++ = '\0';
        }

    return s;
}

void liberer_sequence_mots(sequence_mots *s) {
    if (s->nb_mots) {
        free(s->nb_mots[0]);
        free(s->mots);
    }
    free(s);
}
```