Exercice 1 : Mémoire 1

```c
void g(int **p, int n) {
    **p = n; ②
    *p = &(**p) + 1; ③
}

void f(int* p) {
    g(&p, 0); ④
    g(&p, 1); ⑤
    g(&p, 2);
}

int main() {
    int tab[3]; ①
    f(&tab[0]);
}
```

Q. 1 Représenter la mémoire aux points de programme marqués ci-dessus.

Exercice 2 : Mémoire 2

```c
int* choose(int* p1, int* p2) {
  bool b = rand() % 2 == 0;
  int* res;  ①
  if (b) {
    res = p1;
  } else {
    res = p2;
  }  ②
  return res;
}

void f(int tab[], int tab_len) {
  int* p =choose(&tab[0], &tab[tab_len-1]);    ③
  *p = 0;
}

int main() {
  int tab[3] = {1, 2, 3}; ⓪
  f(tab, 3); ④
}
```

Q. 1 Représenter la mémoire aux points de programme marqués ci-dessus.

Exercice 3 : Mémoire 3

```c
void g(int* p1, int* p2, int ** p) {
  ①
  if (*p1 > *p2) {
    *p = p1;
  } else {
    *p = p2;
  }
}

void f(int tab[4], int* ptab[2], int** p) {
  g(&tab[0], &tab[1], &ptab[0]); ②
  g(&tab[2], &tab[3], &ptab[1]); ③
  g(ptab[0], ptab[1], p); ④
}

int main() {
  int tab[4] = {0, 3, 2, 1};
  int* ptab[2];
  int* p;   ⓪
  f(tab, ptab, &p); ⑤

}
```

Q. 1 Représenter la mémoire aux points de programme marqués ci-dessus.

Exercice 4 : En vrac

Q. 1 Écrire, en C, une fonction prenant en arguments deux pointeurs p et q vers le même tableau d'entiers t, (de sorte que p pointe vers la case du tableau d'indice i et q pointe vers la case d'indice j avec $i \leqslant j$) et retournant l'entier maximale du tableau $t[i..j]$.

Q. 2 Écrire, en C, une fonction prenant en arguments trois tableaux T, P et R de même taille n, tels que T contient des entiers P contient des pointeurs vers des cases du tableau R qui contient des entiers. Cette fonction devra modifier le tableau R de sorte que sa case d'indice i contienne la valeur $T[j]$ si $P[j]$ pointe vers la case d'indice i de R. Votre fonction devra afficher un message d'erreur si cela n'est pas possible.

Q. 3 Écrire, en C sans utiliser de boucle for/tant que, ni de fonction auxiliaire, une fonction `int strlen(char* c)` retournant la longueur d'une chaîne de caractères.

Exercice 5 : Structures 1

Q. 1 Définir un type structuré `date` permettant la représentation d'une date contenant les grandeurs année, mois, jour.

Q. 2 Définir une fonction `inf_date` prenant deux dates en arguments et testant si la première précède la seconde au sens large.

Q. 3 Définir un type structuré `personne` permettant la représentation d'une personne ayant : un nom, une date de naissance et une date de décès.

Q. 4 Définir une fonction `en_vie` prenant en arguments une date et une personne et testant si cette personne est en vie à cette date.

Q. 5 Définir une fonction `rencontre_possible` prenant en arguments deux personnes et testant s'il est possible que ces deux personnes se soient rencontrées.

Q. 6 Définir une fonction `nb_en_vie` prenant en arguments un tableau de personnes et une date et retournant le nombre de personnes dans le tableau qui sont en vie à la date indiquée.

Exercice 6 : Structures 2

Q. 1 Définir un type structuré `points` permettant la représentation d'un point du plan.

Q. 2 Définir un type structuré `rectangle_par` permettant la représentation d'un rectangle dont les côtés sont parallèles aux axes du plan.

Q. 3 Définir une fonction `dans_rectangle` prenant en arguments un point et un rectangle et testant si le point est dans le rectangle.

Q. 4 Définir une fonction `rectangle_intersecte` prenant en arguments deux rectangles et testant s'ils s'intersectent.

Q. 5 Définir une fonction `rectangle_intersection` prenant en arguments deux rectangles qui s'intersectent et calculant l'intersection (un rectangle).

Q. 6 Définir une fonction `translate` prenant en arguments un rectangle et un point P et retournant le rectangle obtenu par translation du rectangle d'entrée du vecteur $\overrightarrow{OP}$.

Exercice 7 : Structures 3

Q. 1 Définir un type structuré `intervalle` permettant la représentation d'un intervalle $[\![a; b]\!]$ de $\mathbb{Z}$ avec $a, b \in \mathbb{Z}^2$.

Q. 2 Définir une fonction `intervalle_intersecte` permettant de tester si deux intervalles s'intersectent.

Q. 3 Définir une fonction `intervalle_intersection` permettant le calcul de l'intersection de deux intervalles.

Q. 4 Définir une fonction `intervalle_union` permettant le calcul de l'union de deux intervalles d'intersection non nulle.

Q. 5 Définir un type structuré `sous_ensembles_z` permettant la représentation d'un ensemble d'entiers. Votre structure contiendra un tableau T de 100 intervalles et un entier x. Ce type structuré représente alors l'ensemble $\bigcup_{i=0}^{x-1}$ `T[i]`.

Q. 6 Définir une constante `vide` représentant l'ensemble vide.

Q. 7 Définir une fonction `ajout` permettant l'ajout d'un intervalle à un ensemble d'entiers. Si l'ensemble d'entiers contient déjà un intervalle qui s'intersecte avec l'intervalle à ajouter, on fera l'union de ces deux intervalles. Sinon on en ajoutera un nouveau.

Exercice 8 : Structures 4

Q. 1 Définir un type structuré `personne` contenant deux champs : une chaîne de caractères qui est le nom de la personne et un identifiant entier unique.

Q. 2 Définir un type structuré `relation` contenant deux champs : `id_a` et `id_b` qui sont des identifiants entiers.

Dans la suite de l'exercice on manipule des tableaux `personne personnes[]` et des tableaux de `relation relations[]` tels que les identifiants dans `relation` sont des identifiants de personnes dans `personnes`. Si le tableau `relations` contient une structure `s` cela indique que les personnes d'identifiants `s.id_a` et `s.id_b` se connaissent.

Q. 3 Définir une fonction prenant en paramètres une chaîne de caractères `str` et un tableau de personnes dont un des noms est `str` et retournant l'identifiant associé dans le tableau de personnes.

Q. 4 Définir une fonction prenant en paramètres deux chaînes de caractères qui sont les noms de deux personnes, un tableau de personnes et un tableau de relations et retournant si oui ou non ces deux personnes se connaissent.

Q. 5 Définir une fonction prenant en paramètres deux chaînes de caractères qui sont les noms de deux personnes, un tableau de personnes et un tableau de relations et retournant si oui ou non ces deux personnes connaissent une même troisième personne.