Cours C avancé : épreuve du 20/5/2025

Question 1

Le code ci-dessous contient une erreur à la ligne 26 :

```c
#include <stdlib.h>

typedef struct {
  int n;
  double *c;
} Vecteur;

Vecteur construit(m, x) {
  Vecteur V;
  V.n = m;
  V.c = (double *) malloc(V.n * sizeof(double));
  for (int i=0; i<m; i++)
    V.c[i] = i*x;
  return V;
}

void detruit(Vecteur V) {
  free(V.c);
}

int main()
{
  Vecteur v = construit(10, 2.0);
  Vecteur w = construit(10, 3.0);

  v = w;

  detruit(v);
  detruit(w);
  return 0;
}
```

L'exécution s'arrête à la ligne 29 avec le message "free(): double free detected".

- Expliquez ce que fait la ligne 26 et montrer pourquoi elle provoque l'erreur à la ligne 29.
- Proposer une façon de corriger le code.

Question 2

Dans le code ci-dessous :

```c
#include <stdio.h>

void f()
{
   int i = 4;
   printf("f : i = %d\n", i);
}

void g()
{
   int j;
   printf("g : j = %d\n", j);
}

int main()
{
   f();
   g();
   return 0;
}
```

Quand on le compile et qu'on l'exécute, ce code affiche:

```
f : i = 4
g : j = 4
```

- Expliquer **en détail** pourquoi la valeur affichée de la variable locale j dans la fonction g, est 4 alors que j n'est pas initialisée dans cette fonction g (indication : dans vos explications, on pourra utiliser la pile (stack en anglais))
- Qu'est-ce qu'on obtiendrait pour la valeur de j si la fonction g est appelée avant la fonction f dans le programme principal ?

Question 3

La fonction insere insère une valeur à la position k d'un vecteur v :

```c
#include <stdlib.h>

typedef struct {
  int n;
  double *c;
} Vecteur;

void insere(Vecteur *v, int k, double x) {
  int i, n_old = v->n, n_new = n_old+1;
  double *c_old = v->c;
  v->n = n_new;
  v->c = (double *) malloc(n_new * sizeof(double));
  for (i=0; i<k; i++)
    v->c[i] = c_old[i];
  v->c[k] = x;
  for(i=k+1; i<n_new; i++)
    v->c[i] = c_old[i-1];
}
```

Cette fonction compile et s'exécute correctement.

- Elle contient cependant une erreur de gestion mémoire.
- Expliquer cette erreur.
- Proposer une façon de la corriger.

Question 4

Les structures ci-dessous peuvent représenter un vecteur (`Vecteur`) et une liste (`Liste`):

```
typedef struct _Vecteur {
    int n;
    double * x;
} Vecteur;

typedef struct _Noeud {
    double valeur;
    struct _Noeud * suivant;
} Noeud;
typedef Noeud * Liste;
```

Pour les types de données `Vecteur`et `Liste`, évaluer le nombre d'opérations élémentaires pour chaque opération ci-dessous :

- modification d'un élément existant d'un.e vecteur/liste
- insertion d'un nouvel élément dans un.e vecteur/liste
- suppression d'un élément existant
- union de 2 structures vecteurs/listes: on veut obtenir un.e nouvel.le liste/vecteur qui contient tous les éléments
- afficher le nombre d'éléments

Justifiez vos choix.

Par opération élémentaire, on entend ici : copie d'une valeur double, modification de pointeur, allocation mémoire.

Question 5

- Expliquer les différences entre les types `union` et `struct`, leurs avantage(s) et inconvénient(s).