Langage C
Partiel du 12 mars 2022, durée 2h15

Documents autorisés : Trois feuilles A4 recto-verso de notes personnelles.

Les téléphones doivent être éteints.
Le sujet comporte 4 pages.

Tâchez d'écrire de façon lisible, avec des indentations et des accolades appropriées permettant de voir la fin de blocs de code (fin de boucles, etc.).

Il est inutile d'écrire les `#include`. Dans touts les exercices vous pouvez utiliser les fonctions du C standard vues en cours ou en TP.

Exercice 1 :

Qu'est-ce que affichent les cinq `printf` du programme suivant :

```c
#include <stdio.h>
#include <string.h>
int main(void){
  int tab[]={ 2, 4, 5, 6, 7, 8, 9, 10, 11 };

  int *pt = &tab[ 7 ];
  int *pu = &tab[ 1 ];

  printf( "diff = %td\n",  pt - pu );
  printf( "valeurs = %d, %d\n",  *pu - 1, *(pu - 1));

  printf( "val = %d\n",  *(tab+2)+2 );

  int *p = &tab[4];

  int l = p[-2] ;
  printf( "l = %d\n", l);

  p[-3] = p[-3] + p[-1];

  *(p-1) = *(p+1) + *(p-1);

  for( int i=0; i < 9; i++)
    printf("[ %d ] = %d, ", i, tab[i]);
  return 0;
}
```

Exercice 2 :

Que affiche le programme suivant :

```c
#include <stdio.h>

void incr( int n ){
  n = n+1;
}

int main(void){
  int k = 1;

  for( int i = 0; i < 10; i++ )
    incr( k );

  printf("k= %d\n", k);
}
```

Exercice 3 :

Dans le programme suivant :

```c
  void add_mult( int a, int b, ??? ){
     ???
   }

  int main( void ){

    int k = 6; int l = 4;

    int s, m;

    add_mult( k, l, ??? );

    printf("somme = %d,  produit= %d\n", s, m);

  }
```

la fonction `add_mult()` est sensée de calculer la somme `a+b` et le produit `a*b`.

(1) Vous devez compléter le code de la fonction `add_mult()` en remplaçant les points d'interrogation par **deux** paramètres manquants et écrire le corps de la fonction pour qu'elle implémente la tâche demandée.

(2) Modifier l'appel à la fonction `add_mult` dans `main()` de telle sorte que `main` affiche la somme et le produit de `k` et `l`. (Aucune d'autre modification n'est autorisée.)

Exercice 4 :

On définit :

```
    enum color { BLUE, RED, YELLOW, GREEN };

    struct article{
      unsigned int id;
      enum color c;
      unsigned int poids;
    };

```

(1) Déclarer la variable `x` de type `struct article` et l'initialiser *au moment de la déclaration* de telle sorte que les champs `poids`, `id`, `c` prennent respectivement valeurs `54`, `12`, `YELLOW`.

(2) Déclarer et *en même temps* initialiser un tableau de deux éléments `struct article`, les valeurs de champs `poids`, `id`, `c` sont respectivement `11`, `12`, `BLUE` pour le premier élément et `22`, `33` `RED` pour le deuxième.

Exercice 5 :

Qu'est-ce que affiche le programme suivant :

```c
#include <stdio.h>
typedef struct{
  int a; int b;
} toto;

toto enigme( toto x ){
  toto y ;
  y.a = x.a + x.b;    y.b = x.a - x.b;

  x.a = -x.a;         x.b = -x.b;
  return y;
}

int main( void ){
  toto e;
  e.a=1; e.b = -9;

  toto d = enigme(e);
  printf( "e.a = %d e.b = %d  d.a = %d d.b = %d\n", e.a, e.b, d.a, d.b);
}
```

Exercice 6 :

On définit :

```c
typedef struct{
  unsigned int len;
  int *val;
} vector;
```

Dans la structure `vector` le champ `val` est un pointeur vers un tableau de `int`, le champ `len` donne le nombre d'éléments de ce tableau.
Écrire la fonction

```
vector vect_creer( unsigned int l, int c[])
```

qui prend en paramètre un tableau `c` de `l` éléments et retourne une structure `vect` où le champ `val` pointe vers un tableau qui est une copie de `c` (et le champ `len` est égal à `l`).
Une solution qui fait copier les éléments de `c` *sans faire une boucle* rapportera plus de points.

Exercice 7 :

La structure `vector` est celle de l'exercice précédent.
Écrire la fonction

```
vector vect_concat( vector a, vector b)
```

qui prend deux structures `vect` et retourne un nouveau objet `vector` qui représente un vecteur dont le nombre d'éléments est `a.len + b.len`.
Le champ `val` du résultat doit pointer vers un vecteurs de qui contient tous les éléments de vecteur représenté pas `a` suivi de tous les éléments de vecteur représenté par `b`.
Par exemple si les champs `val` de `a` et de `b` pointent respectivement vers les vecteurs `[1, 2, 3, 4, 5]` et `[11, 12, 13]` alors le champ `val` du résultat pointera vers le vecteur
`[1, 2, 3, 4, 5, 11, 12, 13]` .
On privilégia une solution qui n'utilise pas de boucles.