CORRECTION CONTROLE CONTINU DE PROGRAMMATION C INFORMATIQUE NIV 1 (2022-2023)

Proposez par : GROUPE GENIUS REPETITION

Par : Mr **Joël_yk**

Exercice 1 : (1,5*8) 12pts

Validez les instructions ou affirmations correctes !

a) Les fonctions **fpt** et **ftab** réalisent la même chose.

Valider, Le résultat des 2 fonctions est le même.

b) int (*pt)[6]=montab1;

Valider, ***pt*** est déclaré comme un pointeur sur un tableau de 6 entiers. Le nom du tableau se comporte comme un pointeur sur le 1er élément. Les types correspondent.

c) La boucle de la fonction ***fpt*** peut remplacer celle de ***ftab***.

Valider, l'opérateur ***[]*** peut être appliqué sur un pointeur.

d) La boucle de la fonction ***ftab*** peut remplacer celle de ***fpt***.

Valider, car une déclaration ***int t[]*** comme paramètre est interprétée comme un pointeur.

e) fpt(&montab1);

Invalider, Le paramètre n'est pas un pointeur d'entier, mais un pointeur de tableau d'entiers. Notez que le compilateur **gcc** détecte cette erreur et génère un warning. Toutefois,

l'adresse étant le tableau, elle correspond à l'adresse du 1er entier. C'est pourquoi à l'exécution le programme se comporte comme attendu.

f) ftab(montab3);

Valider, La déclaration ***int tab[]*** du paramètre est interprétée comme un pointeur. ***montab3*** étant un pointeur, il correspond au paramètre.

g) ftab(*pt);

Valider, ***pt*** étant un pointeur sur un tableau, ****pt*** est le tableau.

h) L'utilisation de l'allocation dynamique est correcte.

Valider, La fonction calloc() a le même rôle que malloc(). Elle permet d'allouer de la mémoire. La différence entre les fonctions calloc() et malloc(), c'est que **calloc() initialise à 0 tous les éléments de la zone mémoire**. **calloc = Clear (memory) ALLOCation**
Il faut faire #include <stdlib.h> pour pouvoir l'utiliser.

Voici son prototype :

```
void* calloc(size_t num_elements, size_t size);
```

Le premier argument est le nombre d'éléments qu'on souhaite pouvoir stocker en mémoire et le deuxième est la taille de ces éléments que l'on obtient avec l'opérateur **sizeof()**.

Exercice 2 : *8pts*

Ecrire un Programme C qui Déterminer si un nombre est un nombre d'Armstrong ou pas.

Principe : Un nombre de Armstrong est un entier positif dont la somme des cubes des chiffres vaut cet entier.

Exemple : 153 = 13 + 53 + 33 = 1 + 125 + 27 = 153. Ainsi 153 est Un Nombre d'Armstrong.

```
#include<stdio.h>
int main()
{
int nbr, a, tmp, somme=0;
printf(" Entrez un nombre: ");
scanf("%d", &nbr);
tmp=nbr;
while(nbr>0)
{
a=nbr%10;
somme=somme+(a*a*a);
nbr=nbr/10;
}
if(tmp==somme)
printf(" %d est un nombre Armstrong",tmp);
else
printf(" %d n'est pas un nombre Armstrong",tmp);
return 0;
}
```


" La persévérance, c'est ce qui rend l'impossible possible, le possible probable et le probable réalisé. "

Bonne chance pour la normale les amis.

Contact WhatsApp : +237 $6_{58}{}^{39}{}_{59}{}^{78}$ | Réaliser Par Joël_Yk .