

Question 1 : Organigrammes.

Soit un programme permettant d'inverser une matrice carré. Le principe de l'inversion est de permuter les éléments d'indice [i][j] et [j][i].

1 a) Veuillez donner la partie initiale de l'organigramme permettant l'introduction de la matrice carrée initiale NxN

1 b) Veuillez donner le code correspondant a la partie de l'organigramme fournie qui effectue l'inversion.

1a) voir labo sur les matrices.

1b)

```
for(i=0 ;i<n ;i++)
{
    for(j=0;j<n;j++)
    {
        temp=tab[i];
        tab[i]=tab[j];
        tab[j]=temp;
    }
}
```

Question 2 : Analyse de code.

Soit le code d'un programme devant effectuer la recherche d'un caractère particulier dans une chaîne de caractères. D'un point de vue conceptuel, ce programme est fonctionnel mais le code comprends des erreurs a l'édition de lien ou des dysfonctionnement éventuels a l'exécution.

```
#include <stdio.h>
#include <string.h>
void main ()
{
    char *tab,c ;      char *tab impossible → char *tab= « bonjour a tous », c ;
strcpy(tab, « bonjour a tous »);
    int longueur,i=0 ;
    longueur=strlen(tab) ;
    printf(« Veuillez introduire le caractère recherché ») ;
    scanf(« %c »,&c) ;
    while(i<longueur)
    {
        if(tab[i] == c)
        {
            printf(«Caractere trouve a l'indice %d\n»,i);
            break;
        }
        i++;
    }
    printf(«recherche terminee\n»);
}
```

Question 3 :

- a) Quelle est la taille d'un pointeur d'entier et d'un pointeur de flottant. De quoi dépend-elle ?

Elle fait 4 octets, soit 32 bits, pour un système avec une architecture 32 bits.

- b) Donner 3 façons différentes de déclarer une structure ainsi que les variables de ce type.

- **déclaration du type:**

```
struct identificateur {  
    type 1    champ 1;  
    type 2    champ 2;  
    type 3    champ 3;  
    .  
    .  
    type N    champ N;  
};
```

- **déclaration de la variable de type structuré:**
 struct identificateur ident_variable;

On peut également retrouver **deux autres syntaxes moins usitées:**

1. On déclare les variables en même temps que la structure en faisant suivre l'accolade fermée des noms des variables.

```
struct identificateur {  
    type 1 champ 1;  
    type 2 champ 2;  
    type 3 champ 3;  
    .  
    .  
    type N champ N;  
}variable1, variable 2,... ;
```

2. Si l'intérêt d'utiliser un identificateur de type ne se fait pas sentir car on ne désire plus par la suite déclarer de variables de ce type, on peut omettre celui-ci. On retrouvera alors la structure suivante:

```
struct {  
    type 1 champ1;  
    type2 champ2;  
    type3 champ 3;  
    .  
    .  
    type N champ N;  
}variable1, variable2,... ;
```

Question 4 :

4.1

En analysant les différentes lignes de codes qui suivent, veuillez expliquer à quoi elles correspondent et veuillez ajouter pour chaque cas quelques lignes supplémentaires permettant d'en assurer une meilleure compréhension.

- a) `printf(« %s\n »,tab[i]->nom) ;`
- b) `if(tab[i]!='c') continue;`
- c) `printf(“%d\n,f(10)); /* f n'est pas une fonction */`

- a) `tab[i]` est considérée comme la variable (c'est à dire qu'on a un tableau de variables de type structuré) et qu'on affiche le champ nom de cette variable.
- b) Si l'instruction est vérifiée, on passe à l'itération suivante (ce qui implique que la ligne de code soit dans une boucle).
- c) `f` n'est pas une fonction, mais une macro, ce qu'on peut se représenter comme étant une fonction mathématique. La ligne de commande affiche un nombre entier qui sera le résultat de la fonction mathématique de `f(x)` défini dans les directives de précompilation ou `x` est remplacé par 10 à chacune de ses occurrence.

4.2

Que pouvez vous dire de l'exécution des lignes de codes suivantes :

Exemple1 :

<code>char buffer[255];</code>	On définit un tableau de 255 cases
<code>char *ptrbuf ;</code>	On définit un pointeur sur une variable de type char
<code>strcpy(buffer, « bonjour a tous ») ;</code>	On copie bonjour à vous dans le tableau.
<code>ptrbuf=buffer ;</code>	On prend l'adresse du premier element de buffer et on la met dans ptrbuf.
<code>printf(« %d\n », sizeof(buffer)) ;</code>	Affiche 255 (taille de buffer en octets)
<code>printf(« %d\n », sizeof(ptrbuf)) ;</code>	Affiche 4 (taille du pointeur en octets)

Exemple2:

```
char tab[10][10];
char *buffer=&tab[1][1];
printf(“%d\n”,buffer[1][1]);
```

Quel sera l'élément de tab qui va s'afficher?

Dans ce cas-ci, il faudrait voir si le compilateur prend la référence `[1][1]` comme une référence `[1]`. Si c'est le cas, je dirais que ça ne prête pas à conséquence et que la réponse affichée est la l'élément `tab[1][1]` mais ce ne serait bon que pour les 10 premiers éléments. Si ce n'est pas le cas, on ne peut présager de la réponse.