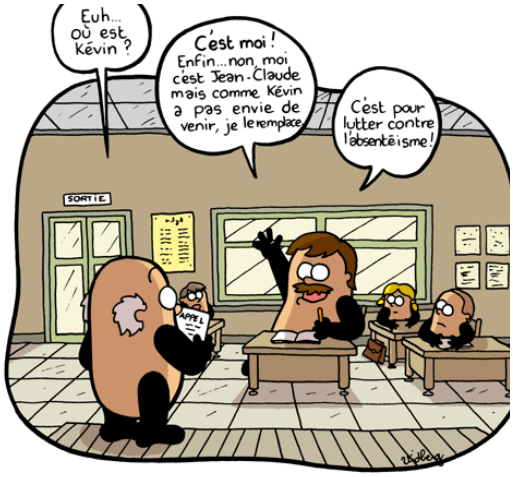

Examen Final Programmation C
Décembre 2013

Remarque :

- **Programmation C :** Exercice 1 à 5, Durée 1h30.
- **Programmation C + Complément d'algo :** Exercice 1 à 6, Durée 2h.
- **Aucun document, pas de smartphone !**

Exercice 1 : printf (5pts)

Pour chacun des exemples suivants, donnez le résultat

1)
```c
char b='a';
b += 3;
printf("%c", b);
```

2)
```c
float a=25.5;
int res;
res = ((int)a) * 4.5 + 0.5 * (float)3 - 100;
printf("%d", res);
```

3)
```c
void accumulator(int b, int c){
    b+=c;
}

main(){
    int i, b=0;
    for(i=0; i<10; i++)
        accumulator(b, i);
    printf("%d",b);
}
```

4)
```c
#define __FCT x*x
typedef float reel;
reel x = 5;
printf("%d", __FCT);
```

5)
```c
unsigned int i, a, b, val=18;

for(i=0, a=1, b=0; i<32; i++, a=a<<1)
    b += (val&a)? 1 : 0;

printf("%d\n",b);
```

Exercice 2 : Que fait ce programme ? (2.5 pts)

Dire ce que fait ce programme

```c
#include<stdio.h>
#define SIZE 10

int whatIsThis( const int b[], int p );

int main(){
    int x;
    int a[SIZE]={1,2,3,4,5,6,7,8,9,10};
    x=whatIsThis(a,SIZE);
    printf("Le resultat est %d\n", x);

    return 0;
}

int whatIsThis(const int b[], int p){
    if (p==1){
        return b[0];
    }else{
        return b[p-1]+whatIsThis(b, p-1);
    }
}
```

Exercice 3 : Portée des variables (2.5 pts)

Donner la portée des éléments suivants (Niveau fonction, fichier, bloc, prototype de la fonction) :

1) Variable ***x*** dans ***main***
2) Variable ***y*** dans ***cube***
3) Fonction ***cube***
4) Fonction ***main***
5) Prototype de la fonction ***cube***
6) Identifiant ***y*** dans le prototype de la fonction ***cube***

```c
#include<stdio.h>

int cube(int y);

int main(){
    int x;

    for(x=1;x<=10;x++)
        printf("%d\n",cube(x));
    return 0;
}

int cube(int y){
    return y*y*y;
}
```

Exercice 4 : Erreurs (2.5 pts)

Trouver les erreurs et corriger les.

1)

```
int g(void){
    printf("Dans la fct g\n");
    int h(void){
        printf("Dans la fct h \n");

    }
}
```

2)

```
int sum(int x, int y){
    int result;
    result = x+y;
}
```

3)

```
int sum(int n){
    if(n==0)
        return0;
    else
        n+sum(n-1);
}
```

4)

```
void f(float a){
    float a;
    printf("%f", a);
}
```

5)

```
void product(void){
    int a, b, c, result;
    printf("Entrer 3 entier:");
    scanf("%d%d%d", &a, &b, &c);
    result = a*b*c;
    printf("Le résultat est %d", result);
    return result;
}
```

Exercice 5 : Structures de donnée (2.5pts)

Soit la structure de données suivante :

```
struct etudiant{
    int id;
    char name[20];
    double gpa;
}
struct etudiant s, *sptr,sarray[10];
```

a) Dire si l'on peut assigner une valeur à ces variables ?

```
1) sptr = ?;
2) s = ?;
3) s.gpa = ?;
4) sarray[3] = ?;
5) sptr->id = ?;
```

b) Trouver les expressions syntaxiquement correctes

```
1) sptr = (struct etudiant *) malloc( 10* sizeof(struct etudiant));
2) s = &sptr;
3) double *dptr = s->gpa;
4) double *dptr = s.gpa;
5) s = (struct etudiant) malloc(1*sizeof(struct etudiant));
```

Exercice 6 : Pile, File (5 pts)

On veut simuler par programme la distribution de cartes à jouer pour un jeu de 52 cartes. Il y a 4 couleurs de cartes (numérotées de 1 à 4), et 13 valeurs de cartes par couleur (numérotées de 1 à 13). On définit le fichier d'en tête **cartes.h** suivant :

```
#ifndef CARTE_H
#define CARTE_H

typedef struct{
    int couleur;
    int valeur;
}Carte;

typdef struct{
    Carte*  premier; // Pointeur sur 1er element de la liste
    Carte*  dernier; // Pointeur sur dernier element de la liste
    int     nbElt;   // Nb d'élément dans la paquet
}PaquetCarte;

typedef PaquetCarte tabJoueur[4];

void    insererEnFinDePaquet    (PaquetCarte* p, int couleur, int valeur);
void    listerCartes            (PaquetCarte* p);
void    creerTas                (PaquetCarte* p);
void    battreLesCartes         (PaquetCarte* p, PaquetCarte* paquetBattu);
void    distribuerLesCartes     (PaquetCarte* p, tabJoueur joueur);

#endif
```

Ecrire les fonctions suivantes :

- `void insererEnFinDePaquet (PaquetCarte* p, int couleur, int valeur);`
qui insère une carte de couleur et de valeur données en fin du paquet p

- `void listerCartes (PaquetCarte* p);`
qui liste la couleur et la valeur des cartes du paquet de cartes p

- `void creerTas (PaquetCarte* p);`
qui crée un paquet de cartes p contenant les cartes dans l'ordre couleur 1 pour les 13 cartes, puis couleur 2 pour les 13 suivantes, etc. soit en tout 52 cartes.

- `void battreLesCartes (PaquetCarte* p, PaquetCarte* paquetBattu);`
qui crée à partir du PaquetCarte p contenant 52 cartes, un nouveau PaquetCarte paquetBattu résultat. On extrait aléatoirement une carte du paquet p pour l'insérer en fin de paquetBattu jusqu'à ce que p soit vide. Utiliser la fonction rand() de génération de nombre aléatoire.

- `void distribuerLesCartes (PaquetCarte* p, tabJoueur joueur);`
qui distribue jusqu'à épuisement de p (52 cartes), une carte à chacun des 4 joueurs.