lundi 19 aout 2013
Sébastien Combéfis

Examen Aout 2013

Prénom : ..... Nom : .....

Formation : ☐ Électronique ☐ Télécom

Vous avez exactement **2h30** pour répondre à toutes les questions de cet examen. Vous n'avez le droit à rien si ce n'est de quoi écrire, ainsi que le formulaire *« Aide-mémoire du langage C »* annoté de manière raisonnable. N'oubliez pas d'écrire vos nom et prénom de manière lisible sur la première page.

Bonne chance !

1 Extraits de code

Pour chacun des extraits de code suivants, vous devez indiquer ce que son exécution affiche à l'écran.

```
int i = 1;
while (i < 5)
{
    i++;
    printf ("%d", i - 1);
}
```

Q1(a) : _____

```
int j;
for (j = 32; j > 0; j /= 2)
{
    printf ("%d;", j);
}
```

Q1(b) : _____

```
int tab[] = {-1, -2, -3, -4, -5};
printf ("Valeur : %d", tab[-tab[tab[1] + 2] / 2]);
```

Q1(c) : _____

2 Reproduction d'escargots

Nous allons étudier l'évolution d'une population d'escargots. Tous les ans, la moitié des escargots en vie meurt. Ensuite, chacun des escargots qui est toujours en vie donne naissance à cinq petits escargots. Par exemple, si on a au départ 4 escargots, après la première année la moitié meurt, il en reste donc 2. Chacun de ces deux va donner naissance à 5 petits et on se retrouve donc avec $2+10=12$ escargots. On continue ensuite avec la même logique. Vous devez écrire une fonction qui calcule combien d'escargots il y aura après "`n`" années, sachant qu'il y a avait "`init`" au début.

```
int populationSize (int init, int n)
{

}
```

3 Lecture de fichier

Soit un fichier texte contenant, sur chacune de ses lignes, une lettre. La fonction suivante, que vous devez compléter, lit le fichier ligne par ligne et compte le nombre de fois que la lettre "`e`" apparait. Elle renvoie ce nombre. La fonction suppose que le fichier existe bel et bien sur l'ordinateur.

```
int countE (char *path)
{
    FILE *file = fopen (path, "r");

    int c;
    int count = 0;
    while ( Q3(a) )
    {
        Q3(b)
    }
    Q3(c)
    return count;
}
```

Q3(a) : _____

Q3(b) :

Q3(c) : _____

4 Mémoire et pointeurs

Soit la variable **int** `**tab` et la situation suivante en mémoire :

	...
`tab` : 2000	3000
	...
3000	5000
	7000
	...
5000	8
	3
	...
7000	16
	7
	-2
	...

Quelles sont les valeurs des expressions suivantes :

1. `tab`
2. `&tab`
3. `*tab`
4. `tab[0]`
5. `tab + 1`
6. `*(tab + 1)`
7. `tab[1][2]`
8. `&(tab[1])`
9. `*(tab + 1) + 2`
10. `*(*(tab + 1) + 2)`

5 Carnet d'adresses

Soit la structure suivante représentant un entrée d'un carnet d'adresses :

```
struct contact {
   char *name;
   int phonenumber;
};
typedef struct contact CONTACT;
```

Écrivez une fonction permettant de créer un nouveau contact, et qui renvoie un pointeur `CONTACT*`.

```
CONTACT* createContact (char *name, int phonenumber)
{









}
```

Écrivez une fonction qui reçoit en paramètre une liste de `CONTACT` (attention, les éléments de la liste ne sont donc pas des pointeurs) de taille `N` et qui recherche s'il y a une personne avec le numéro de téléphone `phonenumber` dans la liste. Si le numéro n'est pas trouvé, la fonction doit renvoyer le pointeur `NULL`.

```
CONTACT* findContact (CONTACT *adressbook, int N, int phonenumber)
{









}
```

Enfin, on s'intéresse à la fonction `main`. Celle-ci a initialisé une liste de deux contacts. Vous devez écrire le reste de la fonction. Elle doit chercher si elle trouve un contact dont le numéro est "`7162712`". Si elle le trouve, elle affiche son nom et si elle ne le trouve pas, elle affiche un message indiquant que le numéro n'a pas été trouvé.

Attention, votre fonction main doit fonctionner même si on change l'initialisation de la liste.

```
int main()
{
    CONTACT *adressbook = malloc (2 * sizeof (CONTACT));
    adressbook[0] = *createContact ("Dylan", 7368271);
    adressbook[1] = *createContact ("Jeremy", 1827317);









}
```