TD5 – Manipulations de pointeurs

Langage C (LC4)

semaine du 1^{er} mars

1 Arithmétique des pointeurs

Question 1. On considère les déclarations suivantes.

```
struct st3 { int a; int b; int c; };
int t[30];
int *p = t;
char *s = (char *) t;
struct st3 *pst3 = (struct st3 *) t;
```

Indiquez les valeurs des expressions ci-dessous après l'exécution de la boucle suivante.

```
int i;
for (i = 0; i < 30; i++)
  t[i] = 10 * i;
```

On supposera qu'on travaille sur une machine où la taille des **`int`** et des **`char`** valent respectivement 4 et 1.

1. `*p + 2`
2. `*(p + 2)`
3. `&p + 1`
4. `&t[4] - 3`
5. `t + 3`
6. `&t[7] - p`
7. `p + (*p - 10)`
8. `*(p + *(p + 8)- t[7])`
9. `s[4]`
10. `&(s[4])- &(s[2])`
11. `pst3[3].b`
12. `((struct st3 *)&t[6])- pst3`

2 Chaînes de caractères

Quelques remarques préliminaires :
- pour stocker une chaîne de n caractères, il est nécessaire de réserver $n+1$ caractères, pour pouvoir stocker le `'\0'` indiquant la fin de la chaine ;
- ne pas confondre :
  - le caractère `'\0'` ;
  - une chaîne vide, qu'on peut noter `""` et qui est représentée en mémoire par une chaîne dont la première case a pour valeur `'\0'` ;
  - la valeur `NULL`, qui peut être affectée à une variable de type `char *` et qui indique qu'elle ne pointe vers aucune zone de mémoire.

Vous implémenterez les fonctions suivantes.

Question 2. `int strlen(const char *)` renvoie le nombre de caractères d'une chaîne. Le `const char *` spécifie que le contenu de la chaîne ne sera pas modifié par la fonction.

Question 3. `int strcmp(const char *s1, const char *s2)` compare deux chaînes `s1` et `s2` : elle renvoie un nombre négatif si la chaîne `s1` est avant `s2` (pour l'ordre lexicographique, l'ordre d'un dictionnaire usuel), 0 si elles sont égales, et un nombre positif sinon.

Question 4. `int palindrome(const char *)` (qui n'est pas dans `<string.h>`) teste si une chaîne est un palindrome, c'est-à-dire une chaîne identique qu'on la lise de la gauche vers la droite ou de la droite vers la gauche (par exemple les mots *laval* ou *subitomotibus*).

Question 5. `char *strchr(const char *s, int c)` renvoie l'adresse de la première occurrence du caractère `c` dans la chaîne `s` en partant du début de la chaîne.

Question 6. `char *strsep(char **stringp, const char *separateurs)` coupe une chaîne en deux à la première occurrence d'un caractère séparateur.
La chaîne à couper est `*stringp`, la chaîne `separateurs` contient tous les caractères séparateurs. Le premier caractère séparateur trouvé est remplacé par `'\0'` dans `*stringp` et la valeur de `*stringp` est modifiée (d'où l'intérêt de passer l'adresse de la chaîne) pour pointer sur le caractère suivant. L'ancienne valeur de `*stringp` est renvoyée.

Question 7. la fonction `char *strcpy(char *dst, const char *src)` copie la chaîne `src` dans `dst`, y compris le caractère de fin de chaîne (et renvoie `dst`). La chaîne `dst` est supposée pointer vers une zone allouée avec une taille suffisante.

Question 8. `char *strcat(char *dst, const char *s)` concatène la chaîne `s` à la suite de `dst` (et renvoie `dst`). Plus précisément, elle écrit par-dessus le caractère `'\0'` à la fin de `dst` puis ajoute un `'\0'` à la fin de la concaténation.
Comme pour `strcpy()`, la chaîne `dst` est supposée pointer vers une zone allouée avec une taille suffisante.

Remarque : la plupart de ces fonctions se trouvent déjà implémentées dans la bibliothèque `<string.h>`.