Programmation Procédurale en Langage C – TD3

Pointeurs : portée des variables, fonctions, tableaux et chaînes de caractères

Exercice 1 : Portée des variables (tordons le cou à quelques idées reçues...)

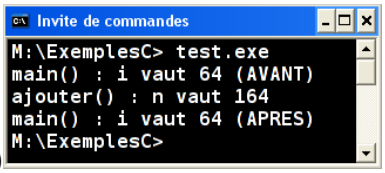
1) Les trois programmes suivants peuvent-ils compiler sans erreur ? Lequel des trois fonctionne correctement ? Que font ces programmes ? Donnez les affichages que produiront leurs exécutions.

```c
/* VERSION 1 */
#include <stdio.h>

void ajouter (int n) {
  n += 100 ;
  printf ("ajouter() : n vaut %d\n", n) ;
}

int main () {
  int i ;

  i = 64 ;
  printf ("main() : i vaut %d (AVANT)\n", i) ;
  ajouter (i) ;
  printf ("main() : i vaut %d (APRES)\n", i) ;

  return 0 ;
}
```

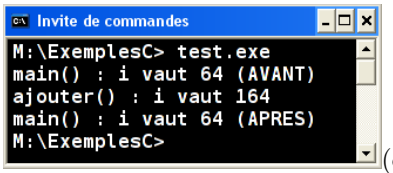
```c
/* VERSION 2 */
#include <stdio.h>

void ajouter (int i) {
  i += 100 ;
  printf ("ajouter() : i vaut %d\n", i) ;
}

int main () {
  int i ;

  i = 64 ;
  printf ("main() : i vaut %d (AVANT)\n", i) ;
  ajouter (i) ;
  printf ("main() : i vaut %d (APRES)\n", i) ;

  return 0 ;
}
```

```c
/* VERSION 3 */
#include <stdio.h>

void ajouter (int n) {
  n += 100 ;
  printf ("ajouter() : n vaut %d\n", n) ;
}

int main () {

  ajouter (64) ;

  return 0 ;
}
```

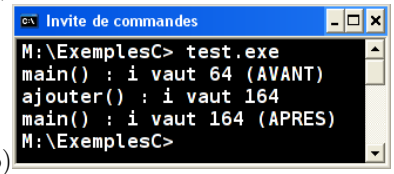
(a)



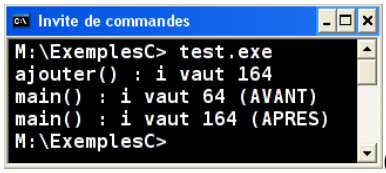
(d)



(b)

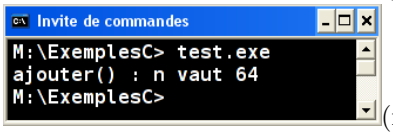


(e)



(c)

(f)



2) Identifiez les variables : globales, locales et statiques.

3) Apportez les modifications nécessaires au bon fonctionnement de la fonction `ajouter()`. Comment devra-t-elle être appelée ? Que devra-t-elle prendre en argument ?

Exercice 2 : Manipulation de pointeurs

1) Que contient `i` à la ligne 6 ?
Que contient `p` à la ligne 8 ?

2) Que vaut `i` à la ligne 7 ? et à la ligne 10 ?
Quels affichages produisent ces lignes ?

3) Expliquez ce qu'il se passe à la ligne 9.

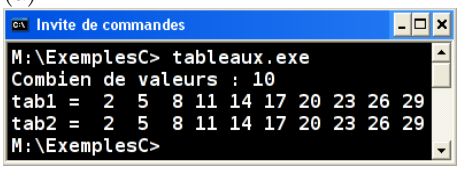
```c
1  #include <stdio.h>
2
3  int main () {
4    int i, *p ;
5
6    i = 10 ;
7    printf ("Le contenu de i vaut %d \n", i) ;
8    p = &i ;
9    *p += 10 ;
10   printf ("Le contenu de i vaut %d \n", i) ;
11
12   return 0 ;
13 }
```

Exercice 3 : Tableaux

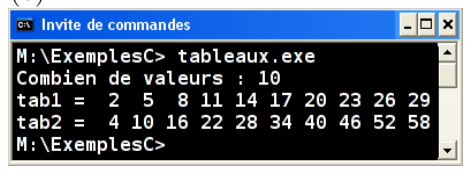
```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main () {
5    int *tab1, *tab2, N, i ;
6
7    /* Recuperer la taille du tableau */
8    printf ("Combien de valeurs : ") ;
9    scanf ("%d", &N) ;
10
11   /* Reserver un espace memoire */
12   tab1 = malloc (N * sizeof(int)) ;
13   if (tab1 == NULL) {
14     fprintf (stderr, "Error: memory allocation\n") ;
15     exit (1) ;
16   }
17
18   /* Initialiser le tableau */
19   for (i=0 ; i<N ; i++)
20     tab1[i] = 3 * i + 2 ;
21
22   /* Copier le tableau */
23   tab2 = tab1 ;
24
25   /* Doubler les valeurs */
26   for (i=0 ; i<N ; i++)
27     tab2[i] = tab2[i] * 2 ;
28
29   /* Afficher le premier tableau */
30   printf ("tab1 = ") ;
31   for (i=0 ; i<N ; i++)
32     printf ("%2d ", tab1[i]) ;
33   printf ("\n") ;
34
35   /* Afficher le deuxieme tableau */
36   printf ("tab2 = ") ;
37   for (i=0 ; i<N ; i++)
38     printf ("%2d ", tab2[i]) ;
39   printf ("\n") ;
40
41   /* Liberer la memoire allouee */
42   free (tab1) ;
43
44   return 0 ;
45 }
```

1) Que fait le programme ? Quel est affichage produit-il ?

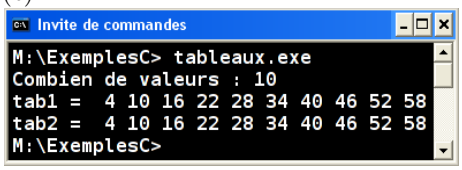
(a)



(b)



(c)



2) Que provoque l'instruction ligne 23 ? Que se passe-t-il en mémoire ?

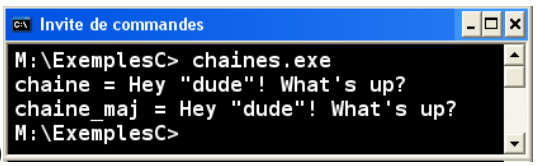
3) Apportez les corrections nécessaires pour obtenir le fonctionnement correct.

Exercice 4 : Chaînes de caractères

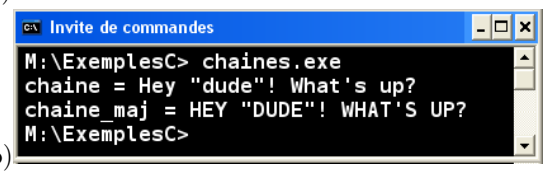
```
1  #include <stdio.h>
2  #include <ctype.h>
3  #include <string.h>
4
5  int main () {
6    char chaine[256] ;
7    char *chaine_maj ;
8    int i ;
9
10   /* Initialiser la chaine de caractere */
11   strcpy (chaine, "Hey \"dude\"! What's up?") ;
12
13   /* Copier la chaine de caractere */
14   chaine_maj = chaine ;
15
16   /* Mettre en majuscule */
17   for (i=0 ; i<strlen(chaine_maj) ; i++)
18     chaine_maj[i] = toupper (chaine_maj[i]) ;
19
20   /* Afficher */
21   printf ("chaine = %s \n", chaine) ;
22   printf ("chaine_maj = %s \n", chaine_maj) ;
23
24   return 0 ;
25 }
```

1) Que fait le programme ?
Quel affichage produit-il ?

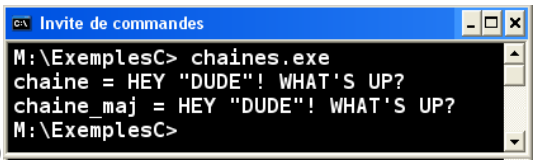
(a)



(b)



(c)



2) Que fait la fonction `strcpy()` ligne 11 ? Que calcule la fonction `strlen()` ligne 17 ?

3) Que provoque l'instruction ligne 14 ? Que se passe-t-il en mémoire ?

4) Comment obtenir le fonctionnement correct ? Quelle fonction de <`string.h`> utiliser ? Quelle fonction faut-il alors appeler en fin de programme ?