Exercice 1

1. Écrire une fonction de prototype `int indice(int t[], n)` qui renvoie un indice i tel que `t[i] = n` si un tel entier existe. Sinon, la fonction renvoie -1.

2. On suppose maintenant que le tableau t est trié par ordre croissant. Il va alors de soi que la recherche d'un indice tel que `t[i] = n` doit pouvoir être accélérée. Implémenter l'algorithme par recherche dichotomique : poser inf=0 et sup=MAX où MAX est le plus grand indice d'un élément du tableau. Tant que $t[\text{sup}] > t[\text{inf}]$, calculer $m = (\text{sup} + \text{inf})/2$, comparer $t[m]$ à n et remettre à jour inf et sup en fonction du résultat. À tout moment, on doit avoir $t[\text{inf}] \leq n \leq t[\text{sup}]$.

Corrigé

Exercice 2

1. Écrire une fonction de prototype `void conv2(int n, int nb2[], int *l)` qui stocke dans le tableau nb2 les bits de l'entier n exprimé en base 2, et qui stoke dans `*l` le nombre de chiffres en base 2 de n auquel on ôte 1. Par exemple, si $n = 6$, qui s'écrit 110 en base 2, l'appel à conv2 aura pour effet nb2[0] = 1, nb2[1] = 1, nb2[2] = 0 et *l=2 (car il y a 3 chiffres dans 110).

2. Écrire une fonction de prototype `int conv10(int nb2[], int l)` qui renvoie l'entier dont les chiffres en base 2 au nombre de $l + 1$ sont stockés dans nb2. On utilisera la factorisation de Horner pour $B = 2$:

$$\sum_{i=0}^{l} a_i B^i = a_0 + B(a_1 + B(\dots))$$

Indication : Soit $n = a_0 2^0 + a_1 2^1 + \cdots + a_l 2^l$ un entier exprimé en base 2. Utiliser la factorisation de Horner revient à calculer les termes de la suite définie par : $u_{l+1} = 0$ et pour tout $0 \leq k \leq l$, $u_k = 2u_{k+1} + a_k$. On a alors $u_0 = n$.

3. Écrire une fonction de prototype `int troisP(int n)` qui renvoit 3^n. La fonction devra utiliser la méthode suivante. Appeler `conv2(n, nb2, &l)` pour calculer les chiffre en base 2 de n puis utiliser la formule suivante :

$$3^n = 3^{a_0 + 2(a_1 + 2(\dots))} = 3^{a_0} \left(3^{a_1 + 2(\dots)}\right)^2$$

4. Essayer de réécrire la fonction précédente sans appel à conv2. C'est-à-dire qu'il faut calculer les chiffres en base 2 au fur et à mesure du calcul de 3^n.

5. Estimer le nombre de multiplications nécessaires pour calculer 3^n par la méthode précédente. Comparer ce nombre au nombre de multiplications utilisées par la méthode naïve.

Corrigé

Exercice 3

1. Écrire une fonction de prototype `int mini(int[])` qui renvoit le plus petit élément du tableau passé en argument.

2. Écrire une fonction de prototype `int indice_mini(int[])` qui renvoit l'indice d'un plus petit élément du tableau.

3. Écrire une fonction de prototype `int trier(int a[], int b[])` qui recopie les éléments du tableau a dans le tableau b, mais en les triant par ordre croissant.

Corrigé

Exercice 4

Dans cet exercice, on définira avec le préprocesseur une constante MAX. Les tableaux qu'on déclarera auront MAX cases.

1. Écrire une fonction de prototype `void permutte(int tab[], int i, int j)` qui permutte les éléments i et j du tableau tab.

2. Écrire une fonction de prototype `int indiceM(int tab[])` qui renvoit le plus petit indice i vérifiant `tab[i] < tab[i-1]`. Si un tel indice n'existe pas, la fonction renverra MAX.

3. Écrire une fonction `void tri(int tab[])` qui trie le tableau tab par ordre croissant. On utilisera l'algorithme suivant : tant que i=indiceM est différent de MAX, permutter les cases i et $i-1$ du tableau.

Corrigé

```
#include<stdio.h>
#define MAX 4


void afftab(int tab[]){
```

```
  int i;

  for(i=0; i<MAX; i++)
    printf("%i %i\n", i, tab[i]);
}


void permutte(int tab[], int i, int j){

  int t;

  t = tab[i];
  tab[i] = tab[j];
  tab[j] = t;
}


int indiceM(int tab[]){

  int i;

  i=0;

  while(i<MAX && tab[i] >= tab[i-1])
    i++;

  return i;
}


void tri(int tab[]){
  int i;

  while ((i=indiceM(tab)) < MAX)
    permutte(tab, i-1, i);
}

main(){
  int t[MAX];

  t[0] = 100;
  t[1] = 90;
  t[2] = 80;
  t[3] = 10;

  //printf("Indice : %i\n", indiceM(t));
  //permutte(t, 3, 6);
```

```
  tri(t);

  afftab(t);
}
```

Exercice 5

1. Écrire une fonction qui donne la longueur d'une chaîne de caractères passée en argument.

2. Écrire une fonction de prototype `int compare (char a[], char b[])` qui renvoit vrai si la chaîne a est placée avant la chaîne b dans l'ordre alphabétique. On suppose que a et b sont des chaîne de lettres minuscules.

3. Reprendre la question précédentes en supposant que a et b sont des chaînes de lettres quelconques (majuscules ou minuscule)

Corrigé

Exercice 6

Écrire une fonction de prototype `convlettre(int n, char lettre[])` qui écrit dans la chaîne lettre le nombre n en écrit en français et en toute lettres. La fonction devra supporter les nombres jusqu'à 999.

Corrigé