Langage C : TD7

Exercice 1 :

On utilise une pile dans laquelle on veut empiler les valeurs entières de 0 à 9, dans cet ordre, mais en autorisant l'action de dépiler une valeur à n'importe quel moment. Cela signifie qu'à chaque fois que l'on fait l'action "empiler" on ajoute dans la pile la valeur suivante (entre 0 et 9). Donner, lorsque cela est possible, une séquence d'opérations (`empiler` et `dépiler`) de manière à ce que les valeurs dépilées respectent l'ordre suivant :

- $4\,3\,2\,1\,0\,9\,8\,7\,6\,5$
  - $5\times$ `empiler` et $5\times$ `depiler`
  - $5\times$ `empiler` et $5\times$ `depiler`
- $4\,6\,8\,7\,5\,3\,2\,9\,0\,1$
  - $5\times$ `empiler` et $1\times$ `depiler`
  - $2\times$ `empiler` et $1\times$ `depiler`
  - $2\times$ `empiler` et $5\times$ `depiler`
  - $1\times$ `empiler` et $1\times$ `depiler`
  - Impossible d'avoir 0 puis 1.
- $2\,5\,6\,7\,4\,8\,9\,3\,1\,0$
  - $3\times$ `empiler` et $1\times$ `depiler`
  - $3\times$ `empiler` et $1\times$ `depiler`
  - $1\times$ `empiler` et $1\times$ `depiler`
  - $1\times$ `empiler` et $2\times$ `depiler`
  - $1\times$ `empiler` et $1\times$ `depiler`
  - $1\times$ `empiler` et $4\times$ `depiler`
- $4\,3\,2\,1\,0\,5\,6\,7\,8\,9$
  - $5\times$ `empiler` et $5\times$ `depiler`
  - $1\times$ `empiler` et $1\times$ `depiler`
  - $1\times$ `empiler` et $1\times$ `depiler`
  - $1\times$ `empiler` et $1\times$ `depiler`
  - $1\times$ `empiler` et $1\times$ `depiler`
  - $1\times$ `empiler` et $1\times$ `depiler`

Exercice 2 :

Dans cet exercice on s'intéresse à une pile gérée via une liste chaînée simple. Dans un premier temps cette pile sera utilisée pour inverser le contenu d'une chaîne de caractères, caractère par caractère. Ainsi on va manipuler une pile contenant des caractères, définie à partir du type suivant :

```c
typedef struct cell{
    char c;
    struct cell *s;
} Pile;
```

Il nous faut tout d'abord implanter les fonctions de manipulation de base d'une pile. Pour cela, vous écrivez une `fonction`.

1. `void init(Pile **p)`, d'initialisation d'une `Pile` vide.

```
void init(Pile **p)
{
    *p = NULL;
}
```

2. `int est_vide(Pile *p)`, permettant de savoir si la `Pile` pointée par p est vide ou non.

```
int est_vide(Pile *p)
{
    return p == NULL;
}
```

3. `void empiler(Pile **p, char c)`, qui ajoute le caractère `c` au sommet de la `Pile` pointée par p, autrement dit en tête de la liste chaînée.

```
void empiler(Pile **p, char c)
{
    Pile *new;
    new = (Pile *) malloc(sizeof(Pile));
    if (new != NULL)
    {
        new -> c = c;
        new -> s = *p;
        *p = new;
    }
}
```

4. `void depiler(Pile **p, char *c)`, pour retirer et récupérer l'élément au sommet de la `Pile` pointée par p. Ainsi le pointeur `c` permet de récupérer l'élément au sommet (pointeur `NULL` si la pile est vide).

```
void depiler(Pile **p, char *c)
{
    Pile *temp = NULL;
    if (!est_vide(*p))
    {
        temp = *p;
        c = (*p) -> c;
        *p = (*p) -> s;
        free(temp);
    }
}
```

Comment utiliser une pile pour résoudre le problème initial, c'est-à-dire inverser les caractères formant une chaîne de caractères ?

5. Écrire une fonction `void inverse(char *s)` qui effectue ce traitement pour la chaîne de caractères `s` en utilisant une `Pile` en variable locale.

```
void inverse(char *s)
{
    Pile **t;
    int i = 0;

    init(t);
```

```
    while (s[i] != '\0')
    {
        empiler(t, s[i]);
        i++;
    }

    for (int j = 0 ; j < i ; j++) depiler(t, s[j]);

    free(t);
}
```

Exercice 3 :

On veut maintenant utiliser une file pour générer la forme binaire des entiers de 1 à n (pour une valeur de n donnée. Ainsi par exemple pour $n = 5$ on souhaite générer la séquence : $1\ 10\ 11\ 100\ 101$.

Pour cela on manipule une file dont chaque cellule contient une chaîne de caractères (pour simplifier on ne gère pas les chaînes de caractères via une allocation dynamique) :

```
typedef struct cell{
char ch [10];
struct cell *suiv;
} Cellule;

typedef struct{
Cellule *tete, *queue;
} File;
```

En supposant disposer des fonctions de manipulation de base d'une telle file `void init(File *f)`, `int est_vide(File f)`, `void enfiler(File *f, char *ch)`, `void defiler(File *f)`, comment procéder pour générer la séquence associée à une valeur de n fournie ? Compléter la fonction `void generation(int n)` ci-dessous pour réaliser ce traitement.

```
void generation(int n)
{
    int i;
    char temp[10];
    File f;

    for (i = 1 ; i <= n ; ++i)
    {

    }
    while (f.tete != NULL) {defiler(&f);}
}
```