Feuille de TD n°5 : piles, files

MPSI Lycée Clemenceau : option informatique

Mai 2025

Exercice 1 : Ecrire une fonction `rotation` qui place le premier élément de la pile en dernière position.

Exercice 2 :

1) Ecrire une fonction `retourne` qui renverse une pile.

2) Ecrire une fonction `split_parite` qui sépare une pile en deux piles (l'une contenant les éléments d'indice pair, l'autre ceux d'indice impair).

Exercice 3 : *Avec l'utilisation de la librairie* ***`Stack`***
On dispose d'une pile d'assiettes bleues ou rouges numérotées disposées dans le désordre. Comment procéder pour former une pile dans laquelle les assiettes bleues sont situées sous les assiettes rouges, mais en faisant en sorte que pour chacune des deux couleurs l'ordre relatif ne soit pas modifié ? (Autrement dit, si l'assiette bleue i est située sous l'assiette bleue j dans la pile initiale, ce sera toujours le cas dans la pile finale.)

On définit les types :

```
# type couleur = Bleue | Rouge and assiette = couleur * int ;;
```

Rédiger une fonction de type `assiette t -> unit` qui range une pile d'assiette en disposant les assiettes bleues sous les assiettes rouges.

Exercice 4 : A l'aide des fonctions primitives du type pile, écrire une fonction `eval` qui réalise, de façon impérative, l'évaluation d'une expression arithmétique postfixée. Celle-ci est codée sous forme d'un tableau dont les éléments sont du type chaine de caractères.

Exercice 5 : On dit qu'une permutation $(a_1\, a_2\, \ldots\, a_n)$ de $(1\, 2\, \ldots\, n)$ peut être engendrée par une pile lorsque il est possible, à partir de la séquence d'entrée $(1\, 2\, \ldots\, n)$ et d'une pile (initialement vide), de produire la séquence de sortie $(a_1\, a_2\, \ldots\, a_n)$ en utilisant les opérations suivantes :

— empiler l'élément suivant de la séquence d'entrée ;

— ou dépiler le sommet de la pile et l'imprimer à l'écran.

Par exemple, si E et D désignent respectivement chacune des deux opérations permises, la permutation (2 3 1) est engendrée par la suite d'opérations : EEDEDD.

1) Parmi les permutations suivantes, lesquelles peuvent être engendrées par une pile ?

$$(3\,1\,2) \quad (3\,4\,2\,1) \quad (4\,5\,3\,7\,2\,1\,6) \quad (3\,5\,7\,6\,8\,4\,9\,2\,10\,1)$$

2) Montrer que s'il existe un triplet $(i, j, k) \in [\![1, n]\!]^3$ tel que $i < j < k$ et $a_j < a_k < a_i$, alors la permutation $(a_1\, a_2\, \ldots\, a_n)$ n'est pas engendrable par une pile.

3) Écrire une fonction `Caml` déterminant si une permutation peut être engendrée par une pile. Dans le cas d'une réponse positive, la fonction affichera la suite d'opérations permettant de la produire. Les permutations seront représentées par le type `int list`.

4) Montrer enfin que toute permutation peut être engendrée à l'aide de deux piles, et rédiger la fonction `Caml` correspondante.

Exercice 6 : Les nombres de Hamming sont les entiers naturels non nuls dont les seuls facteurs premiers éventuels sont 2, 3 et 5 :

$$1;\ 2;\ 3;\ 4\ 5;\ 6;\ 8;\ 9;\ 10;\ 12;\ 15;\ 16;\ 18;\ 20;\ 24;\ 25;\ 27;\ 30;\ \ldots$$

Le but de cet exercice est de les générer de manière croissante. Évidemment, on peut parcourir un à un tous les entiers en testant à chaque fois si ceux-ci sont des entiers de Hamming, mais cette démarche montre vite des limites (songez que le 2000e entier de Hamming est égal à 8 100 000 000 et le 2001e à 8 153 726 976 : il faudrait tester plus de 53 millions de nombres avant d'augmenter notre liste d'un élément !).

On adopte donc la démarche suivante : on utilise trois files f_2, f_3, f_5 contenant initialement le nombre 1, et on suit la démarche suivante :

1. on détermine le plus petit des trois têtes de file, que l'on note k et que l'on imprime à l'écran ;
2. on retire cet élément des files où il se trouve ;
3. on insère en queue des files f2, f3 et f5 les entiers 2k, 3k et 5k.

Vous l'avez compris : cette démarche utilise le fait que tout nombre de Hamming différent de 1 est le produit par 2, 3 ou 5 d'un nombre de Hamming plus petit.

1) Rédiger une fonction Caml permettant l'affichage des n premiers nombres de Hamming.

2) L'inconvénient de la démarche précédente est que le même nombre peut se retrouver dans plusieurs des trois files. Modifier votre fonction pour que cela ne soit plus le cas.