Algorithmique et Programmation 2
Travaux Pratiques – Séance 9

À déposer à la fin de la séance 9 sur Arche

Dans cette séance de TP, nous nous intéressons au type `pile` d'entiers étudié en cours et muni des opérations primitives suivantes :

1. Constructeurs
   - `pile_vide` : ⟶ `pile`
   - `empiler` : `typelt` × `pile` ⟶ `pile`
2. Accès
   - `est_vide` : `pile` ⟶ `booléen`
   - `sommet` : `pile` ⟶ `typelt`
   - `dépiler` : `pile` ⟶ `pile`
3. Axiomes
   - [1] `est_vide(pile_vide) = vrai`
   - [2] `est_vide(empiler(x,P)) = faux`
   - [3] `sommet(pile_vide) = Erreur_typelet`
   - [4] `sommet(empiler(x,P)) = x`
   - [5] `dépiler(pile_vide) = pile_vide`
   - [6] `dépiler(empiler(x,P)) = P`

Dans ce travail, nous implanterons le type abstrait `pile` à l'aide de l'enregistrement suivant :

```c
/* fichier tp9.c */
/* etudiant : nom prenom */

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define TAILLE_MAX 1000
typedef struct Pile pile;
struct Pile{
  int T[TAILLE_MAX];
  unsigned int nb_elements; // nombre d'elements stockes dans la pile
};

/* SIGNATURES DES OPERATIONS PRIMITIVES */
// constructeurs
pile pile_vide () ;
pile empiler (int x, pile P) ;
// acces
bool est_vide (pile P) ;
int sommet (pile P) ;
pile depiler (pile P) ;

int main(int argc, char** argv){

  /* ... */
  return EXIT_SUCCESS;
}
```

Question 1 — Implantation des opérations primitives
Écrivez le code C de chacune des opérations primitives et de leurs procédures de test.

Question 2 —
Il a été vu en cours que l'on pouvait utiliser des piles pour tester qu'une chaîne de caractères était bien parenthésée. En C, le

type `char` étant assimilable à des entiers via le code ascii, on peut coder les piles de caractères à l'aide des piles d'entiers. Le but de cet exercice est d'implanter un tel test :

1. Écrivez et testez une fonction qui teste qu'un caractère est un signe de parenthésage ouvrant : `(`, `[` ou `{`
2. Écrivez et testez une fonction qui teste qu'un caractère est un signe de parenthésage fermant : `)`, `]` ou `}`
3. Écrivez et testez une fonction qui prend deux caractères en entrée et qui teste que le premier est un signe de parenthésage ouvrant, que le second est un signe de parenthésage fermant et que ces deux signes se correspondent : `(` avec `)`, etc.
4. Écrivez et testez une fonction qui teste qu'une chaîne de caractères est bien parenthésée.
5. Parmi les chaînes de caractères mal parenthésées, on distingue :
   (a) Celles qui le sont parce qu'il manque des signes de parenthésage fermant en fin de chaîne, par exemple : `a+(b-[c*d*e/(f+g)]`
   (b) Les autres.

   Écrivez et testez une fonction qui permette de distinguer les chaînes de caractères bien parenthésées (valeur de retour 1), les chaînes de type (a) (valeur de retour -1) et les chaînes de type (b) (valeur de retour -2).
6. ćrivez et testez une fonction qui, à une chaîne de type (a), associe une chaîne de caractères bien parenthésée obtenue en allongeant la chaîne donnée en argument.