DS INFO N°2

Exercice 1 (Questions de cours).

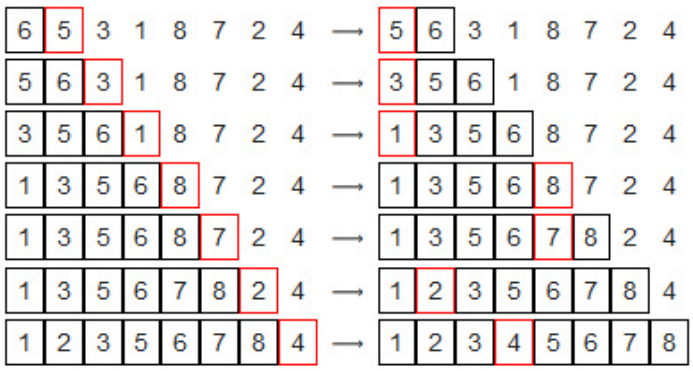
1. (Question de cours.) En n'utilisant que le stylo noir ou le crayon à papier, représenter sur un grand schéma clair et propre la mémoire allouée à un processus en détaillant les différents segments. Ce schéma sert pour la question ci-dessous, lisez donc la suite de l'exercice avant de commencer le schéma.
2. Voici un code C très simple.

```c
#include <stdio.h>
#include <stdlib.h>

int une_fonction(int n, float **ptab)
{
  int p = 2*n + 1;  // FIN DE LA LIGNE 6

  *ptab = calloc(p, sizeof(float)); // FIN DE LA LIGNE 8

  return p;
}

int main(void)
{
  float *tab    = NULL;
  int taille  = une_fonction(3, &tab);

  tab[2] = 2.0*taille; // FIN DE LA LIGNE 18

  if (tab != NULL)
    free(tab);
  tab = NULL;

  return 0;
}
```

3. Donner le diagramme entrée/sortie de l'algorithme codé par la fonction `une_fonction`. Comment appelle-t-on la technique utilisée ? A quoi sert-elle ?
4. On suppose que l'on a suspendu l'exécution de ce programme juste à la fin de la ligne 6. Compléter toujours au crayon ou en noir le schéma de la question précédente en y dessinant les variables allouées par le programme dans les segments **dynamiques** et leur contenu dans leur état **à ce moment de l'exécution**.
5. On exécute l'instruction de la ligne 8. Mettre à jour **en vert** le schéma de la question précédente **à ce moment de l'exécution**. Bien indiquer les relations entre pointeurs et valeurs pointées par des flèches, quand il y en a. Vous pouvez rayer proprement en vert. Pour les valeurs d'adresse vous en "inventerez".
6. On suppose enfin que l'exécution en est maintenant à la fin de la ligne 18. Mettre à jour le schéma avec le stylo **rouge**.

Exercice 2 (Algorithme de tri).

Le tri par insertion est le tri « du joueur de carte ». On parcourt les éléments du tableau. Pour chaque valeur, on va la replacer (l'insérer) au bon endroit parmi le sous tableau de gauche.
Pour cela, on décale les élément à gauche d'un cran vers la droite jusqu'à ce que la case correspondant à la bonne place de la valeur soit libérée.
Puis on passe à la valeur suivante et on la replace au bon endroit dans le sous tableau de gauche...etc et ceci jusqu'à avoir réinséré tous les éléments.
Voici ci-dessous les étapes de l'algorithme du tri par insertion sur le tableau constitué des valeurs 6, 5, 3, 1, 8, 7, 2, et 4, dans cet ordre.



On montre uniquement le résultat de l'insertion de chaque élément sans montrer tous les décalages réalisés.
Par exemple, lors de l'insertion du 2 à l'avant dernière étape, pour placer le 2 au bon endroit, on a décalé vers la droite le 8, puis le 7, puis le 6, puis le 5, puis le 3.

1. On considère le tableau contenant les valeurs suivantes, dans cet ordre : `-11, 8, 4, -3, 1`. En suivant le modèle ci-dessus, détailler les différents états du tableau au cours de l'algorithme
2. Écrire l'algorithme du tri par insertion en pseudo-code **sans utiliser la récursivité** : bien détailler les entrées, les sorties éventuelles et l'algorithme. Vous pouvez utiliser les notations abrégées usuelles pour éviter de longues phrases.
3. Traduire cet algorithme sous la forme d'une fonction OCaml nommée `tri_insertion`
4. En fonction de quel paramètre mesure t-on la complexité de cet algorithme ? Évaluer la complexité temporelle **au pire** de cet algorithme. Vous pourrez annoter proprement votre pseudo-code pour expliquer votre évaluation.
5. Réécrire l'algorithme du tri par insertion en pseudo-code en utilisant cette fois une formulation récursive
6. Quelle structure de données permet de faciliter l'implémentation de cet algorithme en OCaml ? (Réponse en 1 phrase)
7. Traduire cet algorithme récursif en OCaml.

NOM : Prénom :

Exercice 3 (Inférence de type).

Voici plusieurs fonctions OCaml. Indiquer **sur cette page d'énoncé** le prototype de chaque fonction tel qu'inféré par l'interpréteur. Pour chaque fonction, vous expliquerez brièvement comment vous avez inféré le type. Vous pouvez expliquer en annotant directement le code sur l'énoncé. Vous proposerez des annotations claires, concises et efficaces sans y passer trop de temps. Ne rendez QUE cette feuille d'énoncé après y avoir mis votre NOM.

1. Réponse prototype de f1 :

```
let f1 y =
  let tab = [|"m"; "p"; "i"|] in
  let r = ref false in
  for j = 0 to (Array.length tab) -1 do
    if y = tab.(j) then
      r := true
  done;
  !r
;;
```

2. Réponse prototype de f2 :

```
let f2 q r s =
  let h = ref 0 in
  while ( !q < (Array.length s) )
  do
    if (s.(!q) > 2. *. r ) then
      h := !h + 1
  done;
  !h
;;
```

3. Réponse prototype de f3 :

```
let f3 k l f =
  let t = k.(f) in
  if (t > l.(f+1) ) then
    (l.(f) <- l.(f+1);l.(f+1) <- t)
;;
```

4. Réponse prototype de f4 :

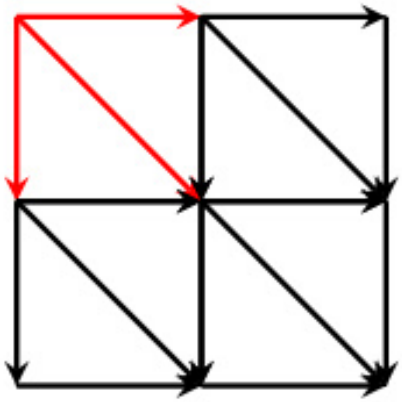
```
let f4 r s t =
  let (_, x) = s in t.(r) <- 2.0 *. x;;
```

5. Réponse prototype de f5 :

```
let f5 k l f =
  let t = k f in
  if (t > l.(f+1) ) then
    (l.(f) <- l.(f+1);l.(f+1) <- t)
;;
```

Exercice 4.

1. Une grille comporte 2×2 cases. Partant du coin supérieur gauche, déterminer le nombre de chemins menant au coin inférieur droit si les seules directions de déplacements autorisées sont celles indiquées par les flèches.



2. Une grille comporte $m \in \mathbb{N}^*$ lignes et $n \in \mathbb{N}^*$ colonnes. On note $c_{i,j}$ le nombre de chemins issus d'un nœud $(i, j) \in [\![0, m]\!] \times [\![0, n]\!]$
Déterminer $c_{0,n}$ et $c_{m,0}$ puis, pour $i \geq 1$ et $j \geq 1$, établir une relation de récurrence donnant $c_{i,j}$.

3. Écrire une fonction OCaml de prototype `c : int → int → int` qui calcule $c_{m,n}$

4. Discuter son efficacité en 3 phrases maximum.

Exercice 5 (Crible d'Ératosthène).

Écrire une fonction en langage C nommée `eratosthene` qui prend en entrée un entier naturel n et qui **affiche** tous les nombres premiers jusqu'à n en utilisant l'algorithme du crible d'Ératosthène. Cet algorithme fonctionne de la manière suivante : on construit un tableau contenant tous les entiers de 2 à n, puis on supprime les multiples de 2, puis ceux de 3, etc. Les entiers qui n'ont pas été barrés sont premiers.
On rappelle qu'il est possible d'utiliser des booléens en C (en utilisant la bibliothèque `stdbool.h`)
On pourra également utiliser la racine carrée qui se nomme `sqrt` (et se trouve dans le module `math.h`)
On ne vous demande d'écrire que la fonction `eratosthene` avec les spécifications données.
Quelle est la complexité temporelle de cet algorithme ?