Devoir surveillé d'informatique

Consignes

- Les programmes demandés doivent être écrits en C ou en OCaml suivant l'exercice. Dans le cas du C, on suppose que les librairies standards usuelles (`<stdio.h>`, `<stdlib.h>`, `<stdbool.h>`) sont déjà importées.
- On pourra toujours librement utiliser une fonction demandée à une question précédente même si cette question n'a pas été traitée.
- Veillez à présenter vos idées et vos réponses partielles même si vous ne trouvez pas la solution complète à une question.
- La clarté et la lisibilité de la rédaction et des programmes sont des éléments de notation.

❑ Exercice 1 : *Recherche des k premiers maximums*

Les fonctions demandées dans cet exercice doivent être écrites en langage C.
On s'intéresse dans cet exercice à la recherche des k premiers maximums d'un tableau de n entiers $t_0 \dots t_{n-1}$ avec $k < n$.

■ Partie I : Cas $k = 1$

1. Dans le cas où $k = 1$, écrire une fonction de signature `int max(int t[], int n)` qui renvoie le premier maximum du tableau `t` de taille `n`.
2. Donner en la justifiant brièvement la complexité de cette fonction.

■ Partie II : Cas général, résolution par un tri

1. On propose de résoudre ce problème en triant le tableau t par ordre décroissant puis en prenant ses k premiers éléments. On suppose déjà écrit une fonction de tri de signature :
   `void int tri(int t[], int n)` qui tri en place le tableau `t` de taille `n` donné en argument. Ecrire une fonction de signature `int * kmax(int t[], int n, int k)` qui utilise cette fonction de tri et renvoie un tableau de taille `k` contenant les `k` premiers maximums de `t`.
2. Donner la complexité de cette fonction en supposant que celle de la fonction de tri est en $O(n \log n)$.

■ Partie III : Cas général, utilisation d'un tas

1. Rappeler la définition d'un tas min binaire.
   Dans la suite on suppose que les tas sont implémentés par la structure de donnée `heap` suivante composée d'un tableau de taille maximale `capacity` et de taille courante `size`.

```c
struct heap
{
    int size;
    int capacity;
    int *tab;
};
typedef struct heap heap;
```

2. Donner l'indice (lorsqu'il existe) du parent de l'élément d'indice i.
3. Donner les indices (lorsqu'ils existent) des fils de l'élément d'indice i.
4. Expliquer le principe de l'insertion d'un élément dans un tas min binaire et donner la complexité de cette opération en la justifiant
5. Même question pour l'extraction du minimum.

   On suppose déjà écrite les fonctions suivantes :

   — `heap make_heap(int cap)` qui renvoie un tas binaire vide de capacité `cap`

   — `bool insert_heap(int nv, heap* mh)` qui insère `nv` dans le tas `*mh`, dans le cas où l'insertion est impossible (tas plein), la fonction renvoie `false`.

— `int getmin(heap * mh)` qui renvoie (en le supprimant du tas) le minimum du tas `*mh`

6. Ecrire une fonction `int* kmax_heap(int t[], int n, int k)` qui renvoie les `k` premiers maximums avec une complexité $O(n \log k)$.