SINF1252 Nom : ..... Prénom : .....

Une queue multithreadée

Utiliser des structures de données dans des programmes multi-threads demande certaines précautions aux programmeurs. En effet, plusieurs threads pourraient tenter d'accéder en même temps à une même structure de données. Pour leur faciliter la tâche, nous vous demandons de concevoir et d'implémenter une file FIFO (First In, First Out) pour ce genre de programmes. Votre file doit pouvoir être manipulée par le biais des fonctions suivantes :

```
/*
 * pqueue_new crée une file avec initialement capacity places.
 */
pqueue pqueue_new (size_t capacity);

/*
 * pqueue_push insère l'élément data à la fin de la file et
 * retourne 0 ssi l'opération réussit. data appartient à l'appelant.
 */
int pqueue_push (pqueue queue, void *data)

/*
 * pqueue_pop place dans data l'élément à la tête de la file et le supprime de
    la file.
 * Retourne 0 ssi l'opération réussit, -1 si la queue est vide.
 * La valeur de data appartient à l'appelé.
 */
int pqueue_pop (pqueue queue, void **data)

/*
 * pqueue_free détruit complètement la file.
 */
void pqueue_free (pqueue queue)
```

La file est implémentée avec un tableau dont la taille varie au fur et à mesure de l'ajout de nouveaux éléments. Elle est implémenté en utilisant un structure opaque pour le programmeur qui est définie comme suit :

```
typedef struct {
size_t capacity;  // capacité actuelle de la queue
size_t n_items;   // nombre d'éléments dans la queue
void *items;      // pointeur vers le tableau contenant la queue
} * pqueue;
```

Proposez une implémentation en C des fonctions `pqueue pqueue_new (void)`, `int pqueue_push (pqueue queue, void *data)` et `int pqueue_get (pqueue queue, void **data)`. Décrivez en français ce qui doit être réalisé par la fonction `void pqueue_free (pqueue queue)` sans l'implémenter en détails.

SINF1252 Nom : . . . . . Prénom : . . . . . .

SINF1252 Nom : ..... Prénom : .....