Voici le code (à compléter) de la fonction `decompte` :

```
int * decompte( int * px )
{
    int y = 0;
    while ( *px > 0 )
      {
          ++y;
          --(*px);
      }
      /* retourner la valeur de y */
}
```

Utilisez correctement les paramètres des fonction `pthread_exit` et `pthread_join` de manière à récupérer la valeur des variables `y` et de les afficher à partir du `main`.

b) Exécutez votre programme pour de grandes valeurs de `x`. Que remarquez-vous ?

c) Si le CPU de votre machine possède plusieurs coeurs, utilisez la commande `taskset` de manière à n'utiliser qu'un seul coeur. Par exemple :

```
$ taskset -c 0 ./mon_programme 100
```

d) Modifiez votre programme de manière à communiquer une deuxième variable de `int` aux deux threads. Appelons cette nouvelles variable `verrou`, identifiez les sections critiques puis utilisez ce `verrou` de la manière suivante :

- Lorsque `verrou` vaut 0, c'est bon, on peut entrer en section critique.
- Lorsque `verrou` vaut 1, c'est que quelqu'un est déjà dans la section critique et qu'il faut attendre que le verrou soit revenu à 0 pour continuer.
- Lorsqu'un processus entre en section critique, il met le `verrou` à 1 et le remet à 0 en quittant la section critique.
- Exécutez votre programme pour de grandes valeur de `x`. Qu'observez-vous ? (Pour abserver un comportement plus *équilibré* vous pouvez ajouter un appel à `pthread_yield` en sortant de la section critique)

e) Corriger votre code de manière à obtenir le résultat attendu.