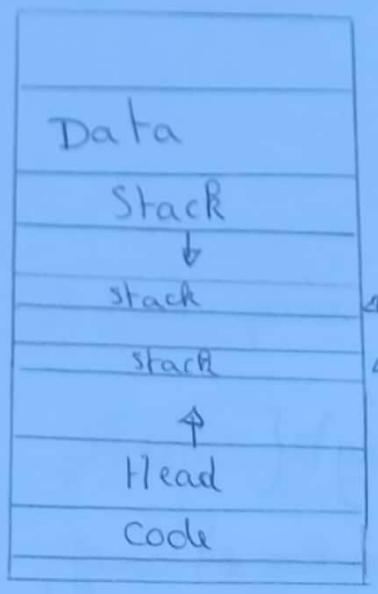
TD3 : Calcul parallèle.



Il y a une stack par thread.

pthread_create -- créer un thread
pthread_join -- attendre un thread

Question 1 :

on met toujours void * quand on crée un thread

```
void * is_prime_wrapper(void * arg).
{
    int n = *(int *)arg
    *(int *) arg = is_prime(n);
    return NULL;
}
```

si on avait eu int is_prime(int, char)

```
{   int n = *(int *) arg
    char i = *(char )((int *) arg + 1);
    *(int *) arg = is_prime(n, i);
    return NULL;
}
```

Un meilleur moyen, si on a ⊕ de fonction est de créer une structure :

```
struct param {
    int n;
    int r;
}

void * is_prime_wrapper(void * arg) {
    struct param p = *(struct param *) arg;
    p->r = is_prime(p->n);
    return NULL;
}
```

Question 2 :

```
int count;
void * isl prime_wrapper (void* arg)
{   Ne fonctionne pas
X   count += is_prime_wrapper(arg->n);
    return NULL;
}
```

cast non fait

Le problème est que l'addition n'est pas atomique, les additions se font en plusieurs étapes.
La valeur n'est pas relu dans la 2ème thread de la variable globale.

Question 3:

```
int main (int argc, char * argv[])
{
    :
    :
    struct param p[N];

    pthread_t   t[N];
    for (i=0; i<N; i++){
        p[i] = is_prime_wrapper
        p[i].n = array[i];
        pthread_create (&t[i], NULL, isprime_wrapper,
                        (void *) &p[i]);

    }
    for (i=0; i<N; i++) {
        pthread_join (t[i], NULL);
        count += p[i].n;
    }
    :
}
```

Question 4:

Il y a ici, beaucoup de thread, alors que les premiers, on fini de s'exécuter.

→ Créer un tableau de taille 8 (par exemple) et on analyse les 8 thread différent.