PDS TD ① du 19/11

2 primitives :

P	up	signal	sem_post
V	down	wait	sem_wait

- sem_post :

1) si le compteur ≤ 0,
on lance 1 processus de la file d'attente et le lance

2) On incrémente le compteur

- sem_wait :

1) si le compteur ≤ 0
on bloque le proc ds la file d'attente

2) On décrémente le compteur

```
P1() {
a1();
sem_post(&sem3);
_____(&sem2);
sem_wait(&sem1);
_____(&sem1);
b1(); }
```

```
P2() {
a2();
sem_post(&sem1);
_____(&sem3);
sem_wait(&sem3);
_____(&sem2);
b2();
}
```

```
P3() {
a3();
sem_post(&sem1);
sem_post(&sem2);
sem_wait(&sem3);
_____(&sem3);
b3(); }
```

```
for( i=0 ; i<N ; i++) sem_init(&sem[i], , 1);
```

```
Pk() {
ak();
for( i=0 ; i<N ; i++){
    if(i != k) sem_post(&sem[i]);
}
for( i=0 ; i<N-1 ; i++) {
    sem_wait(&sem[k]);
}
bk();
}
```

q2

q3

```
sem_init(&barr, , 0);
sem_init(&mutex, , 1);
compteurdea = 0;
N=3; macro
```

```
pk() {
ak();
sem_wait(&mutex);
compteura ++;
if( compteura == N)
    sem_post(&barriere);
sem_post(&mutex);
sem_wait(&barriere);
bk();
sem_post(&barriere); }
```

PDS

TD ② dv. 19/11

Exo 2

```
int main (int argc, char *argv){
    int n, res ; void * p1 ;
    assert (argc != 2) ;
    n = atoi (argv[1]) ;
    pthread_create (&thd, NULL, fib
    (void*) &n) ;
    p1 = (void*) &res ;
    pthread_join( thd, &p1 );
    printf ("%d \n", res ) ;
    return EXIT_SUCCESS ;
```

```
void* fib ( void * arg ){
    int n = *( int * ) arg ; pthread thd [2] ;
    if (n < 2) return n ;
    else {
        int n1 = n-1 ; int n2 = n-2 ;
        int r1, r2 ;
        void * p1, *p2 ;
        p1 = (void *) &r1 ;
        p2 = ____ r2 ;
        pthread ( thd ____ , NULL, fib, (void*) &n1 );
                  -create
        ____ +1 ____ 2);
        pthread_join( thd[0], &p1 ) ;
        ____ 13 — 2) ;
        n = r1 + r2
        return ( void *) &n ;
```

Exo 3

```
#define SIZE 10

int pt_search ( int *tab, unsigned int size, int (*pred)(int))
int i;
if (size < SIZE){for (i=0; i<size; i++) if(pred(tab[i])) return i;
return -1;}
pthread_t thd ; struct arg_s my_arg; int res1; int res2;

my_arg.res = &res1; my_arg.pred = pred; my_arg.start=tab ; my_arg.size=size;

pthread_create (&thd,NULL, pt_search,(void*)&my_arg) ;
// ds res1 j'ai -1 ou l'indice de la première partie
res2 = search ( tab + size/2, size/2 , pred); (*)
if(res1 != -1) return res1;
return res2;

}
void * pt_search (void *arg){
struct arg_s * local =(struct arg_s *) arg ;
*(local -> res)= search (local -> start, local-> size, local -> fct);
return (void*) NULL ;

}

(*) pthread_join ( thd ,(void*) NULL);
```

```
Struct  args_s {
    int  *start;
    unsigned int size;
    int (*fct)(int);
            int *res;
};
```

Exo4:

```
int  unique()
    static int count;
    int i;
    pthread_mutex_lock(&mutex);   -> à initialiser
    i =++count;
    pthread_mutex_unlock(&mutex);
    return i;
}
```