TD n°1

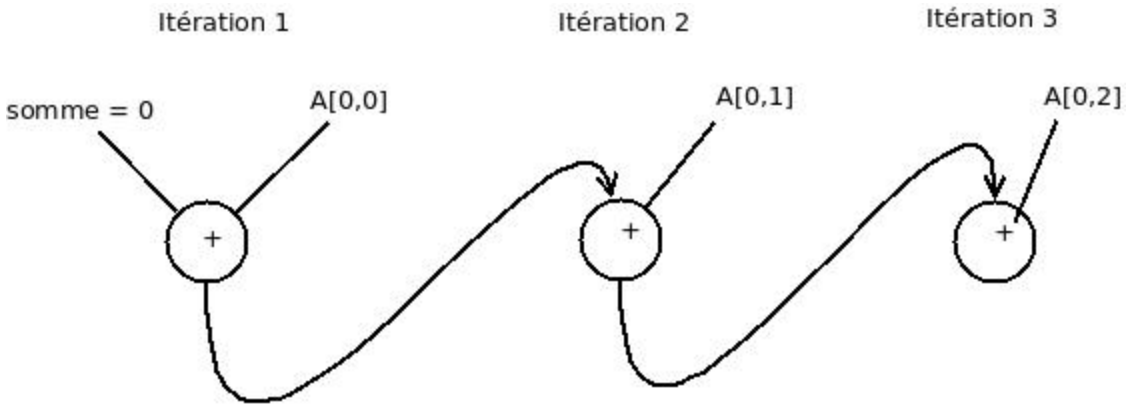
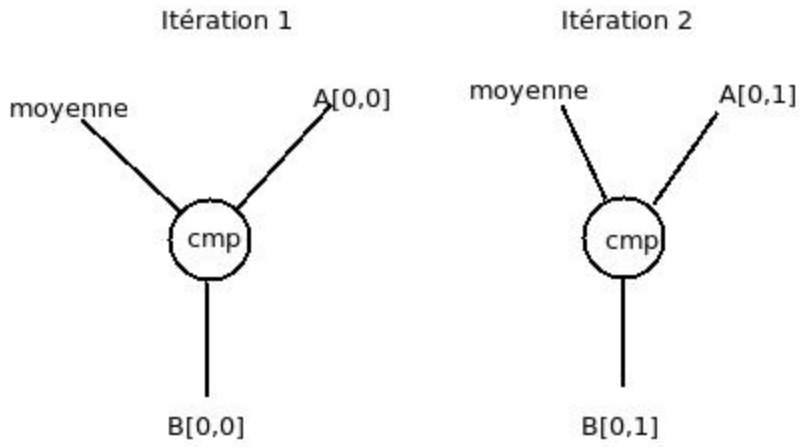
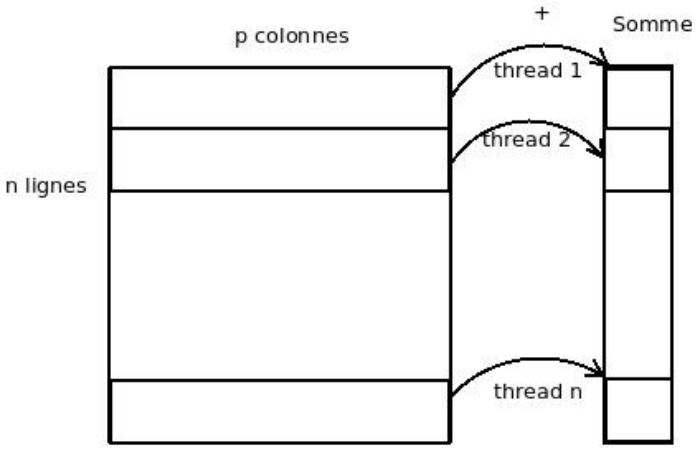
Exercice 1 :

```
int main(){
	int A[N][P], B[N][P] ;
	int somme = 0 ;
	float moyenne ;
	for (int i=0 ; i<N ; i++)
		for (int j=0 ; j<P ; j++)
			somme = somme + A[i][j] ;
	moyenne = somme / (N*P) ;
	for (int i=0 ; i<N ; i++)
		for (int j=0 ; j<P ; j++)
			if (A[i][j] >= moyenne)<
				B[i][j] = 1 ;
			else
				B[i][j] = 0 ;
}
```

Question 1.

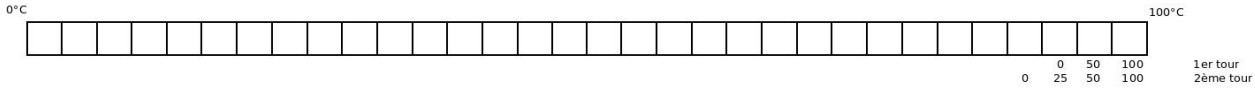
- ```
  int somme = 0;
  for (int i=0; i<N; i++)
  	for (int j=0; j<P; j++)
  		somme = somme + A[i][j];
  ```

  Pas parallélisable : on a besoin de somme de l'itération précédente à chaque itération.

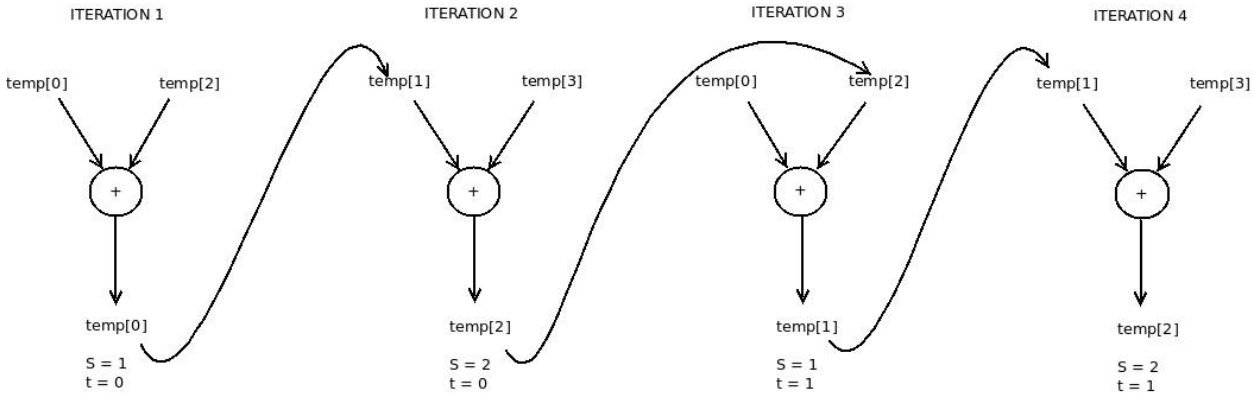

  ```

- ```
  moyenne = somme / (N*P) ;
  for (int i=0 ; i<N ; i++)
  	for (int j=0 ; j<P ; j++)
  		if (A[i][j] >= moyenne)
  			B[i][j] = 1;
  		else
  			B[i][j] = 0;
  ```

  C'est parallélisable



Question 2.

Chaque thread calcule la somme des éléments d'une ligne puis met la valeur dans un tableau.

Une autre boucle va faire la somme des éléments du tableau.



N threads pour calculer les sommes

SYNCHRONISATION : on attend que toutes les sommes soient calculées

1 thread pour la moyenne

N threads reprennent ensuite les boucles suivantes

Pour avoir N/4 threads : chaque thread prend 4 lignes.
  ```

Exercice 2 :

```
int temp[N] = {100, 0, 0, 0, 0, ..., 0};
/* calcul de d'évolution des températures */
for (int t = 0 ; t < TMAX ; t++){
       for (int s = 1 ; s < N-1 ; s++)
              temp[s] = (temp[s-1] + temp[s+1]) / 2;
}
```



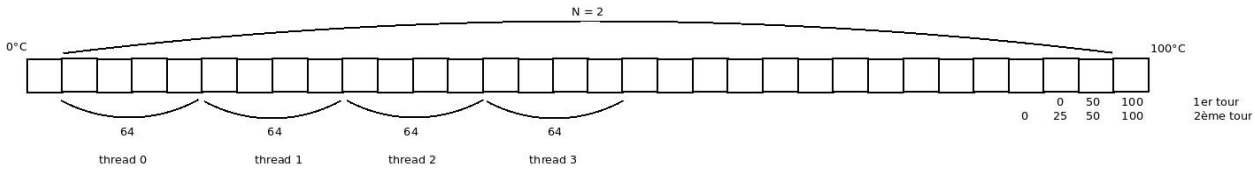
Question 1.



Il y a des dépendances, mais on les ignore. Si on fait suffisamment de tours, on corrigera les erreurs des threads qui ont utilisés les valeurs qui ne sont pas à jour.

Question 2.

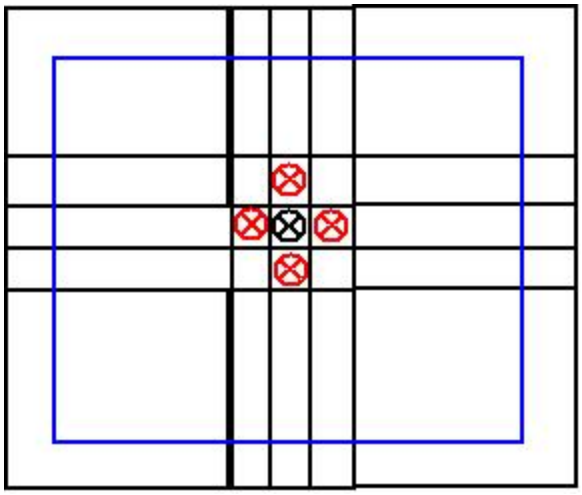
Chaque thread s'occupe de 64 cases



T0
{
temp[1]
…
temp[64]
}
	T1
	{
	temp[65]
	…
	temp[128]
	}
		T2
		{
		temp[129]
		…
		
		}

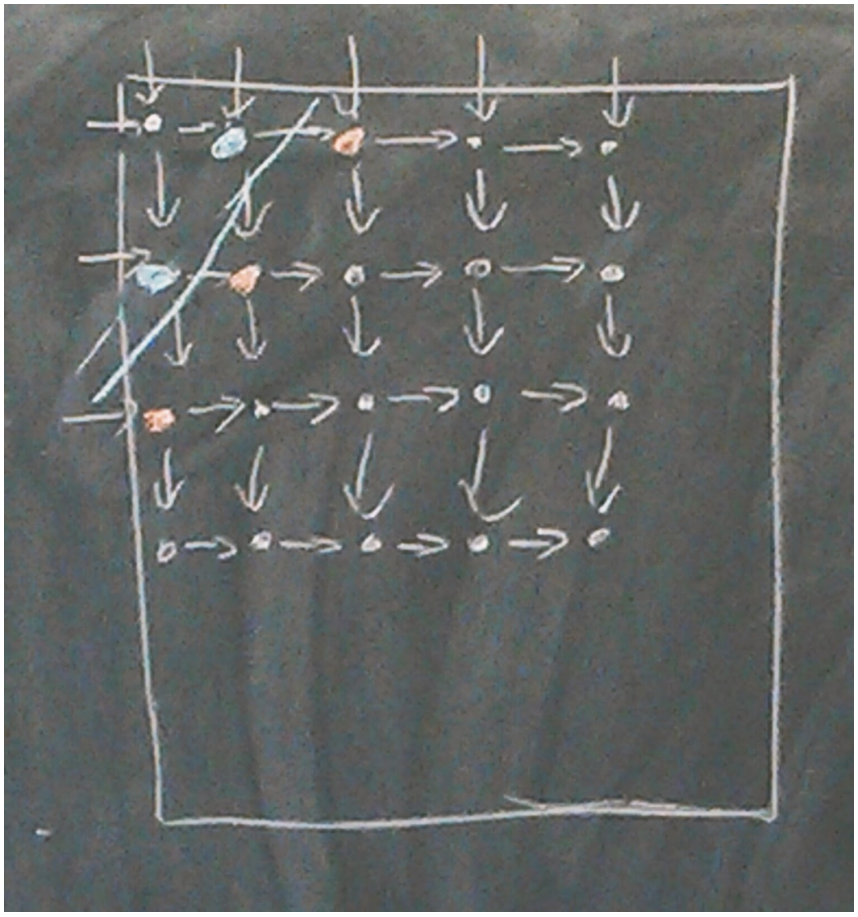
temp[64] a besoin de temp [65] -> Communication

Exercice 3 :



Question 1.

```
for (i=0; i<n; i++) {
        for (j=0; j<n; j++) {
                temp = A[i][j];
                A[i][j] ← 0.2 * (A[i][j]+ A[i][j-1]+
                A[i-1][j]+
                A[i][j+1]+ A[i+1][j]);
                diff ← diff + abs(A[i][j] - temp);
        }
}
```



Découpage en diagonales : 1 thread calcule (0,0) ; 2 threads calculent (0,1) et (1,0) …
⇒ 2N processeurs. Il faut aussi synchroniser à chaque étape (beaucoup d'étapes)
⇒ SOLUTION BIEN MAIS PAS TOP

Question 4.
$n = kp^2$
n largeur et hauteur de la grille
p nombre de processeurs
1 thread par processeur

	Comm.	?
Chaque thread traite n^2/p^2 points		
Chaque thread traite n/p^2 lignes entrelacées	2n	$2n^2/p^2$
Chaque thread traite n/p^2 lignes consécutives	n	2n
Une matrice de n^2/p^2 points		4n/p

$2n^2/p^2 > 2n$