Corrigé du TD de Java n°1

1 FACTORIELLE

```
Factorielle(int n)
{
	result = 1;
	compteur = 1;
	Tant que (compteur <= n)
	{
		result = result * compteur;
		compteur = compteur + 1;
	}
	retourner result;
}
```

2 SATELLITE EN PANNE

1.

```
ValeurAbsolue(int a)
{
	result = 0;
	Si (a > 0)
	{
		retourner a;
	}
	Sinon
	{
		Tant que (a != 0)
		{
			a = a+1;
			result = result+1;
		}
		retourner result;
	}
}
```

2.

```
ValeurAbsolueDifference(int a, int b)
{
      //D'abord, on souhaite placer dans x le minimum entre a et b, et dans y,
le maximum entre a et b
      Si (a < b)
      {
            x = a;
            y = b;
      }
      Sinon
      {
            x = b;
            y = a;
      }

      result = 0;
      Tant que (x < y)
      {
            x = x+1;
            result = result+1;
      }
      retourner result;
}
```

3 RECHERCHER ET TRIER UN TABLEAU D'ENTIERS

Dans la suite, taille sera la taille du tableau. En Java, on peut obtenir la taille d'un tableau **tab** en faisant **tab.length**. Cependant, dans d'autres langages de programmation, ceci n'est pas vrai. Pour cette raison, afin de faire des algorithmes les plus généraux possibles dans le corrigé, je passerai en entrée du programme non seulement le tableau, mais aussi sa taille. Au contrôle, libre à vous de faire comme vous le souhaitez.

1.

```
PositionPlusGrandElement(int tab[], int taille_tab)
{
      max = 0;
      compteur = 1;
      Tant que(compteur < taille_tab)
      {
            Si(tab[max] < tab[compteur])
            {
                  max = compteur;
            }
            compteur=compteur+1;
      }
      retourner max;
}
```

2. Grâce au programme, précédent, ce programme devient très simple à réaliser.

```
PlusGrandElement(int tab[], int taille_tab)
{
      position_max = PositionPlusGrandElement(tab, taille_tab);
      retourner tab[position_max];
}
```

3. Le programme va utiliser le premier programme (voir question 1 de cet exercice) pour trouver où est le plus grand élément du tableau, puis échanger le dernier élément du tableau avec cet élément. Ainsi, le plus grand élément du tableau se retrouve à la fin du tableau.

Puis, on recommence mais en diminuant de 1 la taille du tableau à considérer (on ne veut pas que le programme de recherche du plus grand élément prenne en compte la dernière case de notre tableau).

```
TrierTableau(int tab[], int taille_tab)
{
      taille_a_considerer = taille_tab;
      Tant que (taille_a_considerer > 1)
      {
            position_max = PositionPlusGrandElement(tab, taille_a_considerer);
            //On échange les données des cases position_max et
taille_a_considerer-1
            c = tab[taille_a_considerer – 1];
            tab[taille_a_considerer – 1] = tab[position_max];
            tab[position_max] = c;
            taille_a_considerer = taille_a_considerer – 1;
      }
}
```

4.

```
ChercherElement(int tab[], int taille_tab, int a)
{
      compteur = 0;
      Tant que (compteur < taille_tab)
      {
            Si(tab[compteur] == a)
            {
                  retourner compteur;
            }
            compteur=compteur+1;
      }
      //Si on arrive ici, c'est que le tant que a parcouru tout le tableau sans
trouver notre élément...
      retourner -1;
}
```

5. Ici, on peut s'arrêter de chercher l'élément dès que l'on trouve des valeurs trop grandes dans le tableau.

```
ChercherElement_version2(int tab[], int taille_tab, int a)
{
      compteur = 0;
      Tant que ((compteur < taille_tab) && (tab[compteur]<=a))
      {
            Si(tab[compteur] == a)
            {
                  retourner compteur;
            }
            compteur=compteur+1;
      }
      //Si on arrive ici, c'est que le tant que a parcouru tout le tableau sans
trouver notre élément...
      retourner -1;
}
```