UNIVERSITÉ PARIS-EST

Corrigé du TD de Java n°5

1 IKIGAMI

1. On parcourt le tableau, et toutes les 1000 cases, on met un vaccin mortel.

```
RemplissageVaccin(int Id[], int Death[], int n)
{
	compteur=0;
	Tant que (compteur < n)
	{
		Si (compteur%1000 == 0)
		{
			Death[compteur] = vrai;
		}
		Sinon
		{
			Death[compteur] = faux;
		}
		compteur=compteur+1;
	}
}
```

2. Deux étapes ici : trouver le vaccin du fils, puis le rendre non mortel.

```
RemplissageVaccin(int Id[], int Death[], int n)
{
	compteur=0;
	Tant que (compteur < n)
	{
		Si (compteur%1000 == 0)
		{
			Death[compteur] = vrai;
		}
		Sinon
		{
			Death[compteur] = faux;
		}
		compteur=compteur+1;
	}

	//On rajoute ici le « patch » pour le premier ministre
	compteur=0;
	Tant que (compteur<n)
	{
		Si( (Id[compteur]==1547789546632) && (Death[compteur]==vrai))
		{
			Death[compteur]=faux;
```

```
                  Si(compteur>0)
                  {
                        Death[compteur-1]=vrai; //Pour garder le statistique de
1/1000
                  }
                  Sinon
                  {
                        Death[compteur+1]=vrai;
                  }
            }
            compteur=compteur+1;
      }
}
```

3. Un nombre a se termine par 254 **si et seulement si** a-254 est un multiple de 1000 ou bien vaut 0. Donc, dans tous les cas, on a

a se termine par 254 **si et seulement si** (a-254) modulo 1000 vaut 0.

```
RemplissageVaccin(int Id[], boolean Death[], int n)
{
      compteur=0;
      Tant que (compteur < n)
      {
            Si (compteur%1000 == 0)
            {
                  Death[compteur] = vrai;
            }
            Sinon
            {
                  Death[compteur] = faux;
            }
            compteur=compteur+1;
      }

      //On rajoute ici le « patch » pour le premier ministre
      compteur=0;
      Tant que (compteur<n)
      {
            Si( (Id[compteur]==1547789546632) && (Death[compteur]==vrai))
            {
                  Death[compteur]=faux;
                  Si(compteur>0)
                  {
                        Death[compteur-1]=vrai; //Pour garder le statistique de
1/1000
                  }
                  Sinon
                  {
                        Death[compteur+1]=vrai;
                  }
            }
            compteur=compteur+1;
      }

      //On rajoute ici le « patch » pour la dissidente
      compteur=0;
      Tant que (compteur<n)
      {
            Si( (Id[compteur]-254)%1000 == 0)
            {
```

```
                  Death[compteur]=vrai;
            }
            compteur=compteur+1;
      }
}
```

4. 4 et 2.

5. Idem qu'au TD 1, si ce n'est qu'il faut penser à intervertir en même temps les éléments du tableau Id et du tableau Death quand on les échange.

```
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;
}

TrierTableau(int Id[], boolean Death[], int n)
{
      taille_a_considerer = n;
      Tant que (taille_a_considerer > 1)
      {
            position_max = PositionPlusGrandElement(Id, taille_a_considerer);
            //On échange les données des cases position_max et
taille_a_considerer-1
            c = Id[taille_a_considerer – 1];
            Id[taille_a_considerer – 1] = Id[position_max];
            Id[position_max] = c;

            //On doit aussi échanger les éléments de Death
            c = Death[taille_a_considerer – 1];
            Death[taille_a_considerer – 1] = Death[position_max];
            Death[position_max] = c;

            taille_a_considerer = taille_a_considerer – 1;
      }
}
```

6. Dans tous les cas, on doit, au pire, parcourir toutes les cases du tableau pour trouver un vaccin spécifique. Mais avec la **dichotomie**, on peut faire moins (je vous laisse chercher).