IFT2015 **Audrey Pepin**
Mathieu Perron

Devoir 1

Partie 1:

```
//ce programme décrit une implémentation du type abstrait MTF
EncodeMTF
{
	//Le LinkedList est initialise avant de procéder à encodeMTF
	//Tout entier entré sera entre 0 et M – 1 (donc la liste sera de taille M)
	LinkedList<Integer> permutations = new LinkedList<Integer>()

	for ( var i=0; i < M; i++){
		permutations.add(i);
	}

	encodeMTF(int entier)
	{
		//Trouver l'indice de l'élément
		//indexOf retourne la position de l'élément mis en argument
		int position = permutations.indexOf(entier)
		//Le if( ) sauve quelques étapes si l'élément est déjà à la position 0
		if( position > 0)
		{
			//remove retire l'élément de la liste chaînée à la position mise en
			//argument
			permutations.remove(position)
			//addFirst ajoute l'élément mis en argument au début de la liste chaînée
			//(position 0)
			permutations.addFirst(entier)
		}
		return position;
	}
```

```
        //Méthode decodeMTF
        //Prend une liste de positions encodées avec encodeMTF comme argument
        decodeMTF(Liste<integer> index)
        {
                // initialise une liste de sortie en ordre de 0 a index.length
                permutations = new LinkedList<Integer>()

                for ( int i = 0 ; i < index.length ;  i++ ){
                        permutations.add(i)
                }
                //on prends l'index de chacuns des elements de index
                for(int i = 0; i < index.length; i++)
                {
                        int ind = index[i]
                        //On effectue l'algorithme de MTF sur la liste de permutations
                        permutations.remove(ind)
                        permutations.addFirst(ind)
                }
                //permutations représente maintenant la liste d'entiers qui avait été encodée au
                //départ et qui est maintenant décodée
                return permutations
        }
}
```

```
// méthode récursive d'encodage selon la technique d'encodage omega

    public void encodeOmega(int integer){
        if(integer <= 0){
            return;
        } else {
            // on extrait le nombre de bits de l'entier
            int longueur = bitcount(integer);
            encodeOmega(longueur-1);
            // appel la fonction qui ecrit dans system.out
            ecrire(integer, longueur);
        }
    }

//compte la longueur binaire du nombre
private int bitcount(int n){
        int count = 0;
        while( n >= 1  ){
            n /= 2;
            count++;
        }
        return count;
    }


// cette methode prend l'entier et l'ecris dans un fichier binaire sous system.out
// l'equation utilisant les operateurs binaires est inspiree du code de sedgewick
 private void ecrire(int integer, int size){
        for ( var i=0; i<size; i++) {
            boolean bit = ((integer >>> (size - i - 1)) & 1) == 1;
            writeBit(bit);
        }
    }
```

Partie 2: Résultats

Nom du fichier	Taille originale (en octets)	Taille comprimée selon MTFOmega (en octets)	Taille comprimée selon l'outil mis entre ( ) (en octets)
dickens.txt	28 965 453	28 851 205	(zip) 10 620 011
magna-carta.txt	79 702	78 026	(gzip)19 755
war+peace.txt	3 202 941	3 173 007	(zip) 1 139 144
chromosone4.txt	10 404 835	3 592 817	(gzip)3 118 371
chromosone11.txt	7 134 168	2 477 757	(gzip)2 128 937
ecoli.txt	4 638 690	1 579 238	(7zip) 1 185 233
bible.txt	4 047 392	3 947 686	(7zip) 885 454
pi10millions.txt	10 000 000	6 873 968	(gzip) 4 691 487
Pi1million.txt	1 000 000	687 281	(gzip) 470 449