programmation

examen

L'algorithme LZW pour la compression de fichiers

L'algorithme, publié d'abord par Lempel et Ziv en 1977 a été amélioré en 1984 par Welch. L'idée de base est de remplacer les occurrences de chaînes de caractères par un code. L'algorithme ne fait pas d'analyse préalable du texte, mais range les chaînes de caractères dans un dictionnaire au fur et à mesure qu'elles arrivent.

L'algorithme LZW manipule trois objets différents : le fichier de caractères (le fichier à compresser ou résultat d'une décompression) le fichier de codes (le fichier compressé) et un dictionnaire (la table des chaînes de caractères).

partie 1 : le dictionnaire

Le dictionnaire est un tableau de 4096 chaînes de caractères : les 256 premières chaînes de caractères sont les 256 caractères ; les suivantes seront des sous-chaînes ajoutées lors de la compression ou lors de la décompression.

```
class Dictionnaire{
    private :
        AnsiString rep[4096];
        int nbElem;
    public :
        Dictionnaire();
        Dictionnaire( const Dictionnaire &);
        void ajout ( const AnsiString &a);
        int rang( const AnsiString &a);
        AnsiString operator[]( int i);
        int nbElement(){ return nbElem;}
};
```

Question 1.

Ecrire la définition des constructeurs de la classe *Dictionnaire*. Le constructeur par défaut initialise les 256 premières chaînes avec les 256 caractères et affecte 256 à *nbElem*.

Question 2.

Ecrire la définition de la fonction *void ajout (const AnsiString & a);* qui ajoute la chaîne *a* au dictionnaire.

Question 3.

Ecrire la définition des fonctions *int rang(const AnsiString &a);* qui renvoie le rang de la chaîne *a* dans le dictionnaire (-1 si elle ne s'y trouve pas) et *AnsiString operator[](int i);* qui renvoie la chaîne se trouvant au rang *i* (on supposera que $i < nbElem$).

partie 2 : Compression décompression

L'algorithme de compression est le suivant:

```
chaine = le premier caractère du fichier
tant qu'il y a encore des caractères dans le fichier faire
      caractère = le caractère suivant du fichier
      si( chaîne concaténé à caractère) est dans le dictionnaire
      alors chaîne = chaine concaténé à caractère
      sinon
            ecrire le code de chaîne
            ajouter chaîne concaténé à caractère au dictionnaire
            chaîne = caractère
ecrire le code de chaîne
```

Question 4.

Effectuer l'algorithme précédent sur un fichier en entrée contenant « /WED/WE/WE» On donnera le résultat obtenu dans le fichier de sortie, et l'état du dictionnaire à la fin de l'algorithme.

Question 5.

Ecrire la définition de la fonction *void comprimer(char * entree, char * sortie)* qui comprime le fichier de nom *entree* dans le fichier de nom *sortie*. Le fichier *sortie* est un fichier d'entiers qui sont des codes (rangs dans le dictionnaire) .

L'algorithme de décompression est le suivant:

```
lire le code c1 dans le fichier à décomprimer
ecrire la chaîne de code c1
tant qu'il y a encore des caractères dans le fichier dans le fichier à
décomprimer faire
      lire le code c2 dans le fichier à décomprimer
      chaîne = la chaîne de code c2
      sortir chaîne
      ajouter la chaîne de code c1 + chaîne[1] au dictionnaire
      c1 = c2
```

Question 6.

Effectuer l'algorithme précédent sur un fichier en entrée contenant le résultat obtenu à la question 4.

Question 7.

Ecrire la définition de la fonction *void decomprimer(char * entree, char * sortie)* qui décomprime le fichier de nom *entree* dans le fichier de nom *sortie*. Le fichier *entree* est un fichier d'entiers qui sont des codes (rangs dans le dictionnaire) .