Algoritmos y Estructuras de Datos.
TPL1. 1er Trábajo Práctico de Laboratorio. [2013-08-31]

PASSWD PARA EL ZIP: **`ARV5671MJA`**

Instrucciones

- El examen consiste en que escriban las funciones descriptas más abajo; impleméntandolas en C++ de tal forma que el código que escriban **compile y corra correctamente**, es decir, no se aceptará un código que de algún error de compilación o que tire alguna excepción/señal de interrupción en runtime. Básicamente se hace una evaluación de caja negra, aunque le daremos un rápido vistazo al código.
- Pueden utilizar todas las funciones y utilidades del estándar de C++ que por supuesto contiene a la librería STL.

- **merge-kway**
  Dado un vector de **listas ordenadas** hacer una **fusión K-way**, que hace una fusión de todas las listas en una sola, también ordenada. **`void mergekw(vector<list<int>> &VL, list<int> &L)`**
  Ayuda: El algoritmo K-way consiste en recorrer los primeros elementos de las listas (atención que algunas pueden estar vacías), tomar el menor e insertarlo al fin de la lista L Esto se realiza hasta que todas las listas esten vacías. El algoritmo es similar a la fusión de dos listas, pero generalizado a K listas. Por ejemplo, si $K = 3$ y

  ```
  VL[0]=(1 4 6)
  VL[1]=(1 3 4 6 10)
  VL[2]=(6 8 )
  ```

  Entonces

  ```
  L=(1 1 3 4 4 6 6 6 8 10)
  ```

- **split-list** Dado un vector de listas **`VL1`** y un entero **`M`** debe devolver otro vector de listas **`VL2`** donde las listas tienen a lo sumo M elementos.
  `void split_list(vector<list<int>> &VL1, int M,vector<list<int>> &VL2);`
  Para eso toma cada una de las listas y si su longitud es mayor que **`M`** la va dividiendo en pedazos de longitud **`M`** y la va poniendo en **`VL2`**. Por ejemplo:

  ```
  VL1[0]=(4 6 8 6 4 6 1 3 5 5)
  VL1[1]=(6 4 3)
  VL1[2]=(1 10 3 7 8 1 2 3 8)
  VL1[3]=(6 2 8 8)
  ```

  Entonces si hacemos **`split_list(VL1,3,VL2);`**

  ```
  VL2[0]=(4 6 8)
  VL2[1]=(6 4 6)
  VL2[2]=(1 3 5)
  VL2[3]=(5)
  VL2[4]=(6 4 3)
  VL2[5]=(1 10 3)
  VL2[6]=(7 8 1)
  VL2[7]=(2 3 8)
  VL2[8]=(6 2 8)
  VL2[9]=(8)
  ```

- **join-list** Dado un vector de listas **VL1** escribir una función
  `void join_list(vector<list<int>> &VL1, int M,vector<list<int>> &VL2);`
  que va juntando listas de **VL1** hasta que todas tengan longitud **>=M**.

  ```
  VL1[0]=(2 4 10)
  VL1[1]=(4 8 5)
  VL1[2]=(7 5 7)
  VL1[3]=(2 9)
  VL1[4]=(1 7 5)
  VL1[5]=(3 8 5)
  VL1[6]=(6 3 10)
  VL1[7]=(3 8 4)
  VL1[8]=(2)
  ```

  Entonces si hacemos `join_list(VL1,5,VL2);`

  ```
  VL2[0]=(2 4 10 4 8 5)
  VL2[1]=(7 5 7 2 9)
  VL2[2]=(1 7 5 3 8 5)
  VL2[3]=(6 3 10 3 8 4)
  VL2[4]=(2)
  ```