Curso de Programación en C++.
TPL1. Trabajo Práctico de Laboratorio. [2017-10-02]

PASSWD PARA EL ZIP: FFP1 K25Q 5D4B

Ejercicios

[Ej. 1] **[occurm]**. Dada un vector `v`, y un entero `m`, escribir una función `void occurm(vector<int> &v,int m,vector<int> &vm);` que genere una lista `vm` con todos los elementos que aparecen exactamente `m` veces en `v`. Por ejemplo, si `v=[0,1,2,3,4,5,1,2,4,2]` entonces `occurm(v,2,vm)` debe dejar `vm=[1,4)]` ya que son los únicos dos elementos que están exactamente 2 veces. `occurm(v,1,vm)` debe retornar `vm=[0,3,5]`.

Nota: Los elementos en `vm` deben aparecer en el orden que estaban originariamente (la primera vez) en `v` (concepto de estabilidad).

Ayuda1: Recorrer los elementos de `v`. Para cada elemento de `v` chequear si está en `vm`. Si NO está, contar cuantas veces está en `v` y si el número de veces es el indicado (es decir coincide con `m`) entonces agregarlo al final de `vm`.

Ayuda2: Puede ser conveniente escribir una función `int count(vector<int> &v,int x)` que retorna cuántas veces `x` está contenido en `v`.

[Ej. 2] **[subvmax]** Programar una función `void subvmax(vector<int> &v,vector<int> &vmax);` la cual recibe una lista de enteros `v` y encuentra y retorna el subvector consecutivo `vmax` que obtenga la mayor suma entre todos sus elementos. Notar que debido a que algunos elementos pueden ser negativos el problema no se resuelve simplemente tomando todos los elementos. También es posible que el subvector resultado no contenga ningún elemento, en el caso de que todos los elementos de `v` sean negativos.

Por ejemplo:

```
[1, 2, -5, 4, -3, 2] -> [4]
[5, -3, -5, 1, 7, -2] -> [1,7]
[4, -3, 11, -2] -> [4, -3, 11]
[4  -4  2 2] -> [4]
```

Si hay varios subvectores que den la misma suma máxima debe retornar el primero y el más corto. En el último ejemplo de arriba hay tres subvectores que dan la máxima suma de 4: a saber `[4]`, `[4 -4 2 2]`, `[2 2]`. Por este criterio debe retornar `[4]` ya que es el que empieza primero y el más corto.

Ayuda:

- Escribir una función `int sumsubv(vector<int> &v,int j1,int j2);` que calcula la suma de los elementos del vector `v` en el rango `[j1,j2)`.
- Recorrer todos los pares `[j1,j2)` con `j1<=j2` y calcular el par que tiene la suma máxima.
- Una vez determinado el par `[J1,J2)` que tiene la suma máxima copiar el rango `[J1,J2)` a `vmax`.

[Ej. 3] **[subvector]** Escribir una función predicado `bool subvector(vector<int> &v1, vector<int> &v2);` que determina si `v2` es un subvector contiguo de `v1` es decir si existe un rango `[j1,j2)` tal que `v2=v1[j1,j2)`. Por ejemplo

```
v1=[1 3 2 5 6] v2=[3 2 5] -> true
v1=[1 3 2 5 6] v2=[3 5 6] -> false // No es contiguo
```

Ayuda: Para cada posicicón **j** en **v1** recorrer **v2** y chequear si los elementos de **v2** están en **v1[j,end)**. Si el tamaño de **v1[j,end)** es menor que el de **v2** retornar **false**, si está contenido retornar **true**.

Instrucciones generales

- 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.
- Se incluye un template llamado **program.cpp**. En principio sólo tienen que escribir el cuerpo de las funciones pedidas.
- Hay una función de evaluación, por ejemplo si **f** es la función a evaluar tenemos

  ```
  ev.eval1(f,vrbs);
  ```

  ev.eval1(f,vrbs); toma una serie de casos de prueba de entrada, le aplica la función del usuario **f** y compara la salida del usuario (**user**) con respecto a la esperada (**ref**). Si la verbosidad (el argumento **vrbs**) se pone en uno, entonces la función evaluadora reporta por consola los datos de entrada, la salida de la función de usuario y la salida esperada

mstorti@galileo/texstuff-1.3.2-144-g1e130172/Mon Oct 2 07:22:26 2017 -0300