Algoritmos y Estructuras de Datos.
TPL1. Trabajo Práctico de Laboratorio 1. [2017-09-07]

PASSWD PARA EL ZIP: `6KLD W3AN CC1U`

Ejercicios

ATENCION: Deben necesariamente usar la opción `-std=gnu++11` al compilador, **si no no va a compilar**.

[Ej. 1] **[extract-range]** Dada una lista de enteros `L1` y dos iteradores en la misma `p,q`, extraer el rango `[p,q)` de `L1` en la lista `L2`. Por ejemplo si `L1=(0,4,3,5,2,7,1,8)` y `*p=4`, `*q=1`, entonces `extract_range(L1,p,q,L2)` debe dejar `L1=(0,1,8)`, `L2=(4,3,5,2,7)`.
Nota1: Ambos `p,q` pueden ser `end()`
Nota2: Puede ocurrir que `q` esté antes de `p` en cuyo caso `L2` debe retornar vacío. Por ej.: si `L1=(0,4,3,5,2,7,1,8)` y `*p=7`, `*q=5`, entonces debe quedar: `L1=(0,4,3,5,2,7,1,8)`, `L2=()`.
Consigna: Escribir una función

```
void extract_range(list<int> &L1, list<int>::iterator p,
list<int>::iterator q, list<int> &L2);
```

que realiza la tarea indicada.

[Ej. 2] **[add-elements]** Insertar cada uno de los elementos de la pila `S` en la lista ordenada `L`, la cual debe permanecer ordenada luego de la inserción. La función debe retornar la cantidad de números repetidos en la lista `L` luego de la inserción. Tener en cuenta que si hay más de dos ocurrencias del mismo número, dicho número cuenta una única vez en la suma de elementos repetidos.
Ejemplos:
(ej1): `S=[-3,4], L=[-1,4,4,7] --> L = [-3,-1,4,4,4,7], ret=1`
(ej2): `S=[-3,4], L=[-3,4,7] --> L = [-3,-3,4,4,7], ret=2`.
(ej3): `S=[-3,4], L=[-3,4,7,10,10] --> L = [-3,-3,4,4,7,10,10], ret=3`.
Nota: La cantidad de elementos distintos es independiente de si estos elementos duplicados se originaron al insertar los nuevos elementos o existían previamente o ambas cosas a la vez.
Consigna: Escribir una función `int addElements(list<int>& L, stack<int> &S);`
que realiza la tarea indicada.

[Ej. 3] **[coprimos]** Escribir una función `bool coprimos(list<int> &L);` que retorna `true` si todos los elementos de `L` son coprimos entre sí. Recordemos que dos enteros son coprimos entre sí el único entero que divide a ambos es 1.
Por ejemplo:

- 12,15 no son coprimos ya que 3 es divisor de ambos.
- 10,21 son coprimos ya que no tienen ningún divisor (`>1`) común.

Nota: Todos los enteros `x` en `L` son `x>1`.
Ayuda:

- Escribir una función auxiliar
  `void divisors(int x,list<int> &divs);` que retorna en `divs` los divisores del entero `x`. Para ello recorre todos los enteros en `[2,x]` y los incluye en `divs` si dividen a `x`.
- Escribir una función `bool coprimos(int x,int y);` que chequea si `x,y` son coprimos. Para ello genera los divisores de `x` y los de `y` y verificar si no hay elementos comunes en ambos.
- Recorrer todos los pares de valores `xi,xj` en `L` (con `i!=j`) chequear si son coprimos entre sí.

Instrucciones generales

- El examen consiste en que escriban las funciones descriptas más arriba; 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.
- Salvo indicación contraria 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.
- Para cada ejercicio hay dos funciones de evaluación, por ejemplo si **`f`** es la función a evaluar tenemos

  ```
  ev.eval<j>(f,vrbs);
  hj = ev.evalr<j>(f,seed); // para SEED=123 debe dar Hj=170
  ```

  `j` es el número de ejercicio, por ejemplo para el ejercicio 1 tenemos las funciones (**`eval<1>`** y **`evalr<1>`**). La primera **`ev.eval<j>(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

  ```
  m: 10, k: 3
  T(ref): (10 (7 (4 1) 1) (4 1) 1)
  T(user): (10 (7 (4 1) 1) (4 1) 1)
  EJ1|Caso0. Estado: OK
  ```

- **`ucase`**: Además las funciones **`eval()`** tienen dos parámetros adicionales:
  `Eval::eval(func_t func,int vrbs,int ucase);`
  El tercer argumento 'ucase' (caso pedido por el usuario), permite que el usuario seleccione uno solo de todos los ejercicios para chequear. Por defecto está en **`ucase=-1`** que quiere "hacer todos". Por ejemplo **`ev.eval4(prune_to_level,1,51);`** corre sólo el caso 51.
- **Archivo con casos tests JSON:** Los casos test que corre la función **`eval<j>`** están almacenados en un archivo **`test1.json`** o similar. Es un archivo con un formato bastante legible. Abajo hay un ejemplo. **`datain`** son los datos pasados a la función y **`output`** la salida producida por la función de usuario. **`ucase`** es el número de caso.

  ```
  { "datain": {
    "T1": "( 0 (1 2) (3 4 5 6) )",
    "T2": "( 0 (2 4) (6 8 10 12) )",
    "func": "doble" },
   "output": { "retval": true },
   "ucase": 0 },
  ```

- La segunda función **`evalr<j>`** es el chequeo que llamamos **SEED/HASH**. La clase evaluadora genera una serie de contenedores a partir de la semilla **`seed`**, se los pasa a la función del usuario **`f()`**. Las respuestas de la **`f()`** van siendo procesadas por la función interna de hash que genera un **checksum `H`** de las respuestas. Por ejemplo para el primer ejercicio si **`seed=123`** entonces el checksum es **`H=523`**. Una vez que el alumno termina su tarea se le pedirá que corra la función **`evalr<j>()`** de la clase evaluadora con un valor determinado de la semilla **`seed`** y se comprobará que genere el valor correcto del checksum **`H`**.
  Desde el punto de vista del alumno esto no trae ninguna complicación adicional, simplemente debe llenar el parámetro **`seed`** con el valor indicado por la cátedra, recompilar el programa y correrlo. La cátedra

verificará el valor de salida de **H**.

- En la clase evaluadora cuentan con funciones utilitarias como por ejemplo:
  `void Eval::dump(list <int> &L,string s="")`: Imprime una lista de enteros por **`stdout`**. Nota: Es un método de la clase **`Eval`** es decir que hay que hacer **`Eval::dump(VX);`**. El string **`s`** es un label opcional.
  - **`void Eval::dump(list <int> &L,string s="")`**
- Después del parcial deben entregar el programa fuente (sólo el **`program.cpp`**) renombrado con su apellido y nombre (por ejemplo **`messilionel.cpp`**). Primero el apellido.

TPL1. Trabajo Práctico de Laboratorio 1. [2017-09-07]. TABLA SEED/HASH

```
S=123 -> H1=630 H2=107 H3=357
S=577 -> H1=414 H2=204 H3=666
S=393 -> H1=870 H2=425 H3=146
S=686 -> H1=383 H2=944 H3=115
S=349 -> H1=388 H2=724 H3=228
S=762 -> H1=215 H2=704 H3=577
S=690 -> H1=511 H2=264 H3=327
S=663 -> H1=868 H2=194 H3=527
S=340 -> H1=035 H2=720 H3=284
S=872 -> H1=402 H2=739 H3=291
S=711 -> H1=351 H2=245 H3=126
S=367 -> H1=119 H2=353 H3=844
S=882 -> H1=895 H2=756 H3=214
S=162 -> H1=101 H2=719 H3=994
S=767 -> H1=226 H2=390 H3=939
S=429 -> H1=524 H2=513 H3=953
S=622 -> H1=753 H2=088 H3=058
S=969 -> H1=869 H2=887 H3=129
S=893 -> H1=253 H2=392 H3=873
S=311 -> H1=480 H2=806 H3=888
S=529 -> H1=594 H2=108 H3=369
S=721 -> H1=879 H2=507 H3=863
S=384 -> H1=679 H2=052 H3=799
S=798 -> H1=441 H2=024 H3=698
S=615 -> H1=618 H2=518 H3=663
S=386 -> H1=077 H2=086 H3=675
S=215 -> H1=464 H2=745 H3=970
S=935 -> H1=104 H2=681 H3=952
S=292 -> H1=441 H2=428 H3=802
S=821 -> H1=406 H2=484 H3=505
S=527 -> H1=329 H2=587 H3=514
S=359 -> H1=044 H2=274 H3=560
S=626 -> H1=882 H2=816 H3=364
S=226 -> H1=128 H2=646 H3=713
S=236 -> H1=313 H2=753 H3=622
S=468 -> H1=040 H2=125 H3=952
S=529 -> H1=594 H2=108 H3=369
S=630 -> H1=076 H2=251 H3=917
S=923 -> H1=896 H2=739 H3=952
S=335 -> H1=929 H2=082 H3=874
S=802 -> H1=577 H2=532 H3=644
S=958 -> H1=605 H2=236 H3=072
S=967 -> H1=275 H2=397 H3=530
S=656 -> H1=670 H2=894 H3=633
S=242 -> H1=469 H2=421 H3=427
S=973 -> H1=257 H2=187 H3=244
S=219 -> H1=726 H2=852 H3=765
S=437 -> H1=990 H2=796 H3=528
S=624 -> H1=108 H2=586 H3=753
S=670 -> H1=950 H2=535 H3=718
```