



1. (1 punto) Razonar (en términos de eficiencia de las operaciones) qué TDA elegirías entre los que se indican para:
  - (a) Implementar un **conjunto ordenado de enteros** (con funciones básicas de insertar, borrar, y buscar) de entre: un **ABB** (árbol binario de búsqueda), un **APO** (árbol parcialmente ordenado implementado mediante un **heap**) y un **Vector ordenado**.
  - (b) Implementar un **Diccionario** (con funciones básicas de insertar, borrar y buscar) de entre: una **Tabla Hash abierta**, un **vector ordenado** y una **lista ordenada**
  - (c) Implementar un **conjunto no ordenado de enteros** (con funciones básicas de insertar, borrar y buscar) de entre: un **AVL**, un **vector no ordenado**, y una **cola con prioridad** (implementada con un **heap**).

2. (1 punto) Dadas 2 listas de enteros L1 y L2 implementar una función:

```
bool check_sum (const list<int> & L1, const list<int> & L2)
```

que devuelva true si los elementos de L1 pueden agruparse sumando de forma que se puedan obtener los de L2 sin alterar el orden de los elementos.

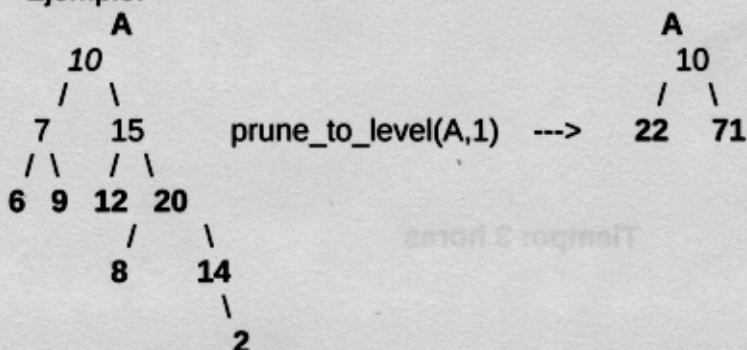
P.ej: Si L1 = {1,2,3,4,1,3,2,5,6,8,3} y L2={ 6, 10, 5, 6, 11} devolvería true

3. (1 punto) Implementar una función

```
void prune_to_level(bintree<int> & A, int level);
```

que pone todos los nodos de un árbol binario A por debajo del nivel *level* sustituyendo la etiqueta de los nodos del nivel más profundo que quede tras podar por la suma de sus descendientes (incluyendo la etiqueta del propio nodo)

Ejemplo:



4. (1 punto) Dado un AVL A, implementar una función, que, dado un nodo *n* en el árbol, devuelva el nodo siguiente a *n* en el recorrido en inorden



5. (1 punto) Detalla cada una de las operaciones siguientes:

- Insertar las claves {8, 16, 12, 41, 10, 62, 27, 65, 13} en una **Tabla Hash cerrada** de tamaño 13. A continuación borrar el 10 y finalmente insertar el valor 51. Resolver las colisiones usando **hashing doble**.
- Construir un **AVL** insertando, en ese orden, las claves anteriores especificando los pasos seguidos e indicando cuando sea necesario el tipo de rotación que se usa.
- Construir un **AVL** con las claves anteriores. Borrar dos elementos en el AVL resultante.

6. (1 punto) Implementar un **iterador** que itere sobre las claves que tengan asociada una lista cuyos elementos sean todos números primos en una clase **Diccionario** definida como:

```
class Diccionario{  
    private:  
        map<int, list<int> > datos;  
        .....  
        .....  
};
```

Han de implementarse (aparte de las de la clase iterador) las funciones **begin()** y **end()**. Se supone implementada una función **bool primo (int x)** que devuelve true si el entero x es primo.

**Tiempo: 3 horas**

- 1.a Heap
- 1.b Tabla hash abierta
- 1.c AVL

```
#include <iostream>
#include <list>

using namespace std;

template <typename T>
void Imprimir(T it_begin, T it_end){
    for (auto it1=it_begin; it1!=it_end; it1++)
        cout << *it1 << " ";
}
```

```
bool check_sum(const list<int> &L1, const list<int>&L2){
    //Inicializamos los iteradores
    list<int>::const_iterator it1 = L1.cbegin();
    list<int>::const_iterator it2 = L2.cbegin();

    //it_anterior sólo lo usamos para comprobaciones
    //no se usa en el algoritmo
    list<int>::const_iterator it_anterior = it1;

    //Recorremos la lista L1 mientras no se produzca un fallo
    bool fallo = false;
    while (it1!=L1.cend() && !fallo){
        //Buscamos una serie consecutiva de elementos de L1
        //que sumen el valor del elemento de L2
        //Sumamos mientras la suma sea distinta del elemento
        //actual de L2, avanzamos it1
        int sum = 0;
        while (sum!=*it2 && it1!=L1.cend()){
            sum += *it1;
            it1++;
        }

        //Salida para comprobaciones
        cout << "Valor en L2: " << *it2
            << " Suma obtenida en L1: " << sum << endl << "Elementos--> ";
        Imprimir(it_anterior, it1);
        cout << endl;
        it_anterior = it1;

        //Al salir del bucle, comprobamos el resultado
        //Si la suma es igual al valor de L2,
        //avanzamos para comprobar el siguiente
        if (sum==*it2)
            ++it2;
        //Si no es igual, hemos terminado
        else
            fallo = true;
    }

    //Al salir del bucle, comprobamos
    //si se ha producido algún fallo
    if (fallo || it2!=L2.cend())
        return false;
    //Si hemos llegado al final de L2 y no hay fallos
    //la comprobación ha sido correcta
    else
        return true;
}
```

```
int main(){
    list<int> L1={1,2,3,4,1,3,2,5,6,8,3};
    list<int> L2={6,10,5,6,11};

    if (check_sum(L1,L2))
```

```
cout<<endl << "Los elementos de L2 sí se obtienen a partir de L1"<<endl;
else
    cout<<endl<<"Los elementos de L2 no se obtienen a partir de L1"<<endl;
}
```

```

#include <iostream>
#include "bintree.h"
using namespace std;

int Suma(bintree<int>::node n){
    if (n.null())
        return 0;
    else
        return *n + Suma(n.left()) + Suma(n.right());
}

void prune_to_level(bintree<int> &A,int level){
    queue<pair<bintree<int>::node,int> >micola;
    pair<bintree<int>::node, int> p(A.root(),0);
    pair<bintree<int>::node, int> aux;
    bintree<int> destino;

    //Insertamos en la cola el pair
    //correspondiente a la raíz (con su nivel, 0)
    micola.push(p);

    //Mientras queden elementos en la cola
    while (!micola.empty()){
        //Tomamos el primer elemento de la cola
        //y lo extraemos
        p = micola.front();
        micola.pop();
        //Si es del nivel a podar
        if (p.second==level ){
            //Sustituimos su etiqueta por la suma del subárbol
            *(p.first)=Suma(p.first);
            //y podamos sus subárboles izquierdo y derecho
            A.prune_left(p.first,destino);
            A.prune_right(p.first,destino);
        }
        //Si no es del nivel a podar
        else{
            //Si tiene hijo izquierdo
            if (!p.first.left().null()){
                //Insertamos el pair de ese nodo en la cola
                aux.first = p.first.left();
                aux.second = p.second+1;
                micola.push(aux);
            }
            //Si tiene hijo derecho
            if (!p.first.right().null()){
                //Insertamos el pair de ese nodo en la cola
                aux.first = p.first.right();
                aux.second = p.second+1;
                micola.push(aux);
            }
        }
    }
}

int main(){

    // Ejemplo:
    //          A
    //          10
    //          /   \
    //          7   15
    //          / \   / \
    //          6   9  12  20
    //          / \   / \
    //          8   14  16  22
    //          / \   / \
    //          10  18  24  26
    //          / \   / \
    //          12  14  26  28
    //          / \   / \
    //          14  16  28  30
    //          / \   / \
    //          16  18  30  32
    //          / \   / \
    //          18  20  32  34
    //          / \   / \
    //          20  22  34  36
    //          / \   / \
    //          22  24  36  38
    //          / \   / \
    //          24  26  38  40
    //          / \   / \
    //          26  28  40  42
    //          / \   / \
    //          28  30  42  44
    //          / \   / \
    //          30  32  44  46
    //          / \   / \
    //          32  34  46  48
    //          / \   / \
    //          34  36  48  50
    //          / \   / \
    //          36  38  50  52
    //          / \   / \
    //          38  40  52  54
    //          / \   / \
    //          40  42  54  56
    //          / \   / \
    //          42  44  56  58
    //          / \   / \
    //          44  46  58  60
    //          / \   / \
    //          46  48  60  62
    //          / \   / \
    //          48  50  62  64
    //          / \   / \
    //          50  52  64  66
    //          / \   / \
    //          52  54  66  68
    //          / \   / \
    //          54  56  68  70
    //          / \   / \
    //          56  58  70  72
    //          / \   / \
    //          58  60  72  74
    //          / \   / \
    //          60  62  74  76
    //          / \   / \
    //          62  64  76  78
    //          / \   / \
    //          64  66  78  80
    //          / \   / \
    //          66  68  80  82
    //          / \   / \
    //          68  70  82  84
    //          / \   / \
    //          70  72  84  86
    //          / \   / \
    //          72  74  86  88
    //          / \   / \
    //          74  76  88  90
    //          / \   / \
    //          76  78  90  92
    //          / \   / \
    //          78  80  92  94
    //          / \   / \
    //          80  82  94  96
    //          / \   / \
    //          82  84  96  98
    //          / \   / \
    //          84  86  98  100
    //          / \   / \
    //          86  88  100 102
    //          / \   / \
    //          88  90  102 104
    //          / \   / \
    //          90  92  104 106
    //          / \   / \
    //          92  94  106 108
    //          / \   / \
    //          94  96  108 110
    //          / \   / \
    //          96  98  110 112
    //          / \   / \
    //          98 100 112 114
    //          / \   / \
    //          100 102 114 116
    //          / \   / \
    //          102 104 116 118
    //          / \   / \
    //          104 106 118 120
    //          / \   / \
    //          106 108 120 122
    //          / \   / \
    //          108 110 122 124
    //          / \   / \
    //          110 112 124 126
    //          / \   / \
    //          112 114 126 128
    //          / \   / \
    //          114 116 128 130
    //          / \   / \
    //          116 118 130 132
    //          / \   / \
    //          118 120 132 134
    //          / \   / \
    //          120 122 134 136
    //          / \   / \
    //          122 124 136 138
    //          / \   / \
    //          124 126 138 140
    //          / \   / \
    //          126 128 140 142
    //          / \   / \
    //          128 130 142 144
    //          / \   / \
    //          130 132 144 146
    //          / \   / \
    //          132 134 146 148
    //          / \   / \
    //          134 136 148 150
    //          / \   / \
    //          136 138 150 152
    //          / \   / \
    //          138 140 152 154
    //          / \   / \
    //          140 142 154 156
    //          / \   / \
    //          142 144 156 158
    //          / \   / \
    //          144 146 158 160
    //          / \   / \
    //          146 148 160 162
    //          / \   / \
    //          148 150 162 164
    //          / \   / \
    //          150 152 164 166
    //          / \   / \
    //          152 154 166 168
    //          / \   / \
    //          154 156 168 170
    //          / \   / \
    //          156 158 170 172
    //          / \   / \
    //          158 160 172 174
    //          / \   / \
    //          160 162 174 176
    //          / \   / \
    //          162 164 176 178
    //          / \   / \
    //          164 166 178 180
    //          / \   / \
    //          166 168 180 182
    //          / \   / \
    //          168 170 182 184
    //          / \   / \
    //          170 172 184 186
    //          / \   / \
    //          172 174 186 188
    //          / \   / \
    //          174 176 188 190
    //          / \   / \
    //          176 178 190 192
    //          / \   / \
    //          178 180 192 194
    //          / \   / \
    //          180 182 194 196
    //          / \   / \
    //          182 184 196 198
    //          / \   / \
    //          184 186 198 200
    //          / \   / \
    //          186 188 198 202
    //          / \   / \
    //          188 190 198 204
    //          / \   / \
    //          190 192 198 206
    //          / \   / \
    //          192 194 198 208
    //          / \   / \
    //          194 196 198 210
    //          / \   / \
    //          196 198 198 212
    //          / \   / \
    //          198 200 198 214
    //          / \   / \
    //          200 202 198 216
    //          / \   / \
    //          202 204 198 218
    //          / \   / \
    //          204 206 198 220
    //          / \   / \
    //          206 208 198 222
    //          / \   / \
    //          208 210 198 224
    //          / \   / \
    //          210 212 198 226
    //          / \   / \
    //          212 214 198 228
    //          / \   / \
    //          214 216 198 230
    //          / \   / \
    //          216 218 198 232
    //          / \   / \
    //          218 220 198 234
    //          / \   / \
    //          220 222 198 236
    //          / \   / \
    //          222 224 198 238
    //          / \   / \
    //          224 226 198 240
    //          / \   / \
    //          226 228 198 242
    //          / \   / \
    //          228 230 198 244
    //          / \   / \
    //          230 232 198 246
    //          / \   / \
    //          232 234 198 248
    //          / \   / \
    //          234 236 198 250
    //          / \   / \
    //          236 238 198 252
    //          / \   / \
    //          238 240 198 254
    //          / \   / \
    //          240 242 198 256
    //          / \   / \
    //          242 244 198 258
    //          / \   / \
    //          244 246 198 260
    //          / \   / \
    //          246 248 198 262
    //          / \   / \
    //          248 250 198 264
    //          / \   / \
    //          250 252 198 266
    //          / \   / \
    //          252 254 198 268
    //          / \   / \
    //          254 256 198 270
    //          / \   / \
    //          256 258 198 272
    //          / \   / \
    //          258 260 198 274
    //          / \   / \
    //          260 262 198 276
    //          / \   / \
    //          262 264 198 278
    //          / \   / \
    //          264 266 198 280
    //          / \   / \
    //          266 268 198 282
    //          / \   / \
    //          268 270 198 284
    //          / \   / \
    //          270 272 198 286
    //          / \   / \
    //          272 274 198 288
    //          / \   / \
    //          274 276 198 290
    //          / \   / \
    //          276 278 198 292
    //          / \   / \
    //          278 280 198 294
    //          / \   / \
    //          280 282 198 296
    //          / \   / \
    //          282 284 198 300
    //          / \   / \
    //          284 286 198 304
    //          / \   / \
    //          286 288 198 308
    //          / \   / \
    //          288 290 198 312
    //          / \   / \
    //          290 292 198 316
    //          / \   / \
    //          292 294 198 320
    //          / \   / \
    //          294 296 198 324
    //          / \   / \
    //          296 298 198 328
    //          / \   / \
    //          298 300 198 332
    //          / \   / \
    //          300 302 198 336
    //          / \   / \
    //          302 304 198 340
    //          / \   / \
    //          304 306 198 344
    //          / \   / \
    //          306 308 198 348
    //          / \   / \
    //          308 310 198 352
    //          / \   / \
    //          310 312 198 356
    //          / \   / \
    //          312 314 198 360
    //          / \   / \
    //          314 316 198 364
    //          / \   / \
    //          316 318 198 368
    //          / \   / \
    //          318 320 198 372
    //          / \   / \
    //          320 322 198 376
    //          / \   / \
    //          322 324 198 380
    //          / \   / \
    //          324 326 198 384
    //          / \   / \
    //          326 328 198 388
    //          / \   / \
    //          328 330 198 392
    //          / \   / \
    //          330 332 198 396
    //          / \   / \
    //          332 334 198 400
    //          / \   / \
    //          334 336 198 404
    //          / \   / \
    //          336 338 198 408
    //          / \   / \
    //          338 340 198 412
    //          / \   / \
    //          340 342 198 416
    //          / \   / \
    //          342 344 198 420
    //          / \   / \
    //          344 346 198 424
    //          / \   / \
    //          346 348 198 428
    //          / \   / \
    //          348 350 198 432
    //          / \   / \
    //          350 352 198 436
    //          / \   / \
    //          352 354 198 440
    //          / \   / \
    //          354 356 198 444
    //          / \   / \
    //          356 358 198 448
    //          / \   / \
    //          358 360 198 452
    //          / \   / \
    //          360 362 198 456
    //          / \   / \
    //          362 364 198 460
    //          / \   / \
    //          364 366 198 464
    //          / \   / \
    //          366 368 198 468
    //          / \   / \
    //          368 370 198 472
    //          / \   / \
    //          370 372 198 476
    //          / \   / \
    //          372 374 198 480
    //          / \   / \
    //          374 376 198 484
    //          / \   / \
    //          376 378 198 488
    //          / \   / \
    //          378 380 198 492
    //          / \   / \
    //          380 382 198 496
    //          / \   / \
    //          382 384 198 500
    //          / \   / \
    //          384 386 198 504
    //          / \   / \
    //          386 388 198 508
    //          / \   / \
    //          388 390 198 512
    //          / \   / \
    //          390 392 198 516
    //          / \   / \
    //          392 394 198 520
    //          / \   / \
    //          394 396 198 524
    //          / \   / \
    //          396 398 198 528
    //          / \   / \
    //          398 400 198 532
    //          / \   / \
    //          400 402 198 536
    //          / \   / \
    //          402 404 198 540
    //          / \   / \
    //          404 406 198 544
    //          / \   / \
    //          406 408 198 548
    //          / \   / \
    //          408 410 198 552
    //          / \   / \
    //          410 412 198 556
    //          / \   / \
    //          412 414 198 560
    //          / \   / \
    //          414 416 198 564
    //          / \   / \
    //          416 418 198 568
    //          / \   / \
    //          418 420 198 572
    //          / \   / \
    //          420 422 198 576
    //          / \   / \
    //          422 424 198 580
    //          / \   / \
    //          424 426 198 584
    //          / \   / \
    //          426 428 198 588
    //          / \   / \
    //          428 430 198 592
    //          / \   / \
    //          430 432 198 596
    //          / \   / \
    //          432 434 198 600
    //          / \   / \
    //          434 436 198 604
    //          / \   / \
    //          436 438 198 608
    //          / \   / \
    //          438 440 198 612
    //          / \   / \
    //          440 442 198 616
    //          / \   / \
    //          442 444 198 620
    //          / \   / \
    //          444 446 198 624
    //          / \   / \
    //          446 448 198 628
    //          / \   / \
    //          448 450 198 632
    //          / \   / \
    //          450 452 198 636
    //          / \   / \
    //          452 454 198 640
    //          / \   / \
    //          454 456 198 644
    //          / \   / \
    //          456 458 198 648
    //          / \   / \
    //          458 460 198 652
    //          / \   / \
    //          460 462 198 656
    //          / \   / \
    //          462 464 198 660
    //          / \   / \
    //          464 466 198 664
    //          / \   / \
    //          466 468 198 668
    //          / \   / \
    //          468 470 198 672
    //          / \   / \
    //          470 472 198 676
    //          / \   / \
    //          472 474 198 680
    //          / \   / \
    //          474 476 198 684
    //          / \   / \
    //          476 478 198 688
    //          / \   / \
    //          478 480 198 692
    //          / \   / \
    //          480 482 198 696
    //          / \   / \
    //          482 484 198 700
    //          / \   / \
    //          484 486 198 704
    //          / \   / \
    //          486 488 198 708
    //          / \   / \
    //          488 490 198 712
    //          / \   / \
    //          490 492 198 716
    //          / \   / \
    //          492 494 198 720
    //          / \   / \
    //          494 496 198 724
    //          / \   / \
    //          496 498 198 728
    //          / \   / \
    //          498 500 198 732
    //          / \   / \
    //          500 502 198 736
    //          / \   / \
    //          502 504 198 740
    //          / \   / \
    //          504 506 198 744
    //          / \   / \
    //          506 508 198 748
    //          / \   / \
    //          508 510 198 752
    //          / \   / \
    //          510 512 198 756
    //          / \   / \
    //          512 514 198 760
    //          / \   / \
    //          514 516 198 764
    //          / \   / \
    //          516 518 198 768
    //          / \   / \
    //          518 520 198 772
    //          / \   / \
    //          520 522 198 776
    //          / \   / \
    //          522 524 198 780
    //          / \   / \
    //          524 526 198 784
    //          / \   / \
    //          526 528 198 788
    //          / \   / \
    //          528 530 198 792
    //          / \   / \
    //          530 532 198 796
    //          / \   / \
    //          532 534 198 800
    //          / \   / \
    //          534 536 198 804
    //          / \   / \
    //          536 538 198 808
    //          / \   / \
    //          538 540 198 812
    //          / \   / \
    //          540 542 198 816
    //          / \   / \
    //          542 544 198 820
    //          / \   / \
    //          544 546 198 824
    //          / \   / \
    //          546 548 198 828
    //          / \   / \
    //          548 550 198 832
    //          / \   / \
    //          550 552 198 836
    //          / \   / \
    //          552 554 198 840
    //          / \   / \
    //          554 556 198 844
    //          / \   / \
    //          556 558 198 848
    //          / \   / \
    //          558 560 198 852
    //          / \   / \
    //          560 562 198 856
    //          / \   / \
    //          562 564 198 860
    //          / \   / \
    //          564 566 198 864
    //          / \   / \
    //          566 568 198 868
    //          / \   / \
    //          568 570 198 872
    //          / \   / \
    //          570 572 198 876
    //          / \   / \
    //          572 574 198 880
    //          / \   / \
    //          574 576 198 884
    //          / \   / \
    //          576 578 198 888
    //          / \   / \
    //          578 580 198 892
    //          / \   / \
    //          580 582 198 896
    //          / \   / \
    //          582 584 198 900
    //          / \   / \
    //          584 586 198 904
    //          / \   / \
    //          586 588 198 908
    //          / \   / \
    //          588 590 198 912
    //          / \   / \
    //          590 592 198 916
    //          / \   / \
    //          592 594 198 920
    //          / \   / \
    //          594 596 198 924
    //          / \   / \
    //          596 598 198 928
    //          / \   / \
    //          598 600 198 932
    //          / \   / \
    //          600 602 198 936
    //          / \   / \
    //          602 604 198 940
    //          / \   / \
    //          604 606 198 944
    //          / \   / \
    //          606 608 198 948
    //          / \   / \
    //          608 610 198 952
    //          / \   / \
    //          610 612 198 956
    //          / \   / \
    //          612 614 198 960
    //          / \   / \
    //          614 616 198 964
    //          / \   / \
    //          616 618 198 968
    //          / \   / \
    //          618 620 198 972
    //          / \   / \
    //          620 622 198 976
    //          / \   / \
    //          622 624 198 980
    //          / \   / \
    //          624 626 198 984
    //          / \   / \
    //          626 628 198 988
    //          / \   / \
    //          628 630 198 992
    //          / \   / \
    //          630 632 198 996
    //          / \   / \
    //          632 634 198 1000
    //          / \   / \
    //          634 636 198 1004
    //          / \   / \
    //          636 638 198 1008
    //          / \   / \
    //          638 640 198 1012
    //          / \   / \
    //          640 642 198 1016
    //          / \   / \
    //          642 644 198 1020
    //          / \   / \
    //          644 646 198 1024
    //          / \   / \
    //          646 648 198 1028
    //          / \   / \
    //          648 650 198 1032
    //          / \   / \
    //          650 652 198 1036
    //          / \   / \
    //          652 654 198 1040
    //          / \   / \
    //          654 656 198 1044
    //          / \   / \
    //          656 658 198 1048
    //          / \   / \
    //          658 660 198 1052
    //          / \   / \
    //          660 662 198 1056
    //          / \   / \
    //          662 664 198 1060
    //          / \   / \
    //          664 666 198 1064
    //          / \   / \
    //          666 668 198 1068
    //          / \   / \
    //          668 670 198 1072
    //          / \   / \
    //          670 672 198 1076
    //          / \   / \
    //          672 674 198 1080
    //          / \   / \
    //          674 676 198 1084
    //          / \   / \
    //          676 678 198 1088
    //          / \   / \
    //          678 680 198 1092
    //          / \   / \
    //          680 682 198 1096
    //          / \   / \
    //          682 684 198 1100
    //          / \   / \
    //          684 686 198 1104
    //          / \   / \
    //          686 688 198 1108
    //          / \   / \
    //          688 690 198 1112
    //          / \   / \
    //          690 692 198 1116
    //          / \   / \
    //          692 694 198 1120
    //          / \   / \
    //          694 696 198 1124
    //          / \   / \
    //          696 698 198 1128
    //          / \   / \
    //          698 700 198 1132
    //          / \   / \
    //          700 702 198 1136
    //          / \   / \
    //          702 704 198 1140
    //          / \   / \
    //          704 706 198 1144
    //          / \   / \
    //          706 708 198 1148
    //          / \   / \
    //          708 710 198 1152
    //          / \   / \
    //          710 712 198 1156
    //          / \   / \
    //          712 714 198 1160
    //          / \   / \
    //          714 716 198 1164
    //          / \   / \
    //          716 718 198 1168
    //          / \   / \
    //          718 720 198 1172
    //          / \   / \
    //          720 722 198 1176
    //          / \   / \
    //          722 724 198 1180
    //          / \   / \
    //          724 726 198 1184
    //          / \   / \
    //          726 728 198 1188
    //          / \   / \
    //          728 730 198 1192
    //          / \   / \
    //          730 732 198 1196
    //          / \   / \
    //          732 734 198 1200
    //          / \   / \
    //          734 736 198 1204
    //          / \   / \
    //          736 738 198 1208
    //          / \   / \
    //          738 740 198 1212
    //          / \   / \
    //          740 742 198 1216
    //          / \   / \
    //          742 744 198 1220
    //          / \   / \
    //          744 746 198 1224
    //          / \   / \
    //          746 748 198 1228
    //          / \   / \
    //          748 750 198 1232
    //          / \   / \
    //          750 752 198 1236
    //          / \   / \
    //          752 754 198 1240
    //          / \   / \
    //          754 756 198 1244
    //          / \   / \
    //          756 758 198 1248
    //          / \   / \
    //          758 760 198 1252
    //          / \   / \
    //          760 762 198 1256
    //          / \   / \
    //          762 764 198 1260
    //          / \   / \
    //          764 766 198 1264
    //          / \   / \
    //          766 768 198 1268
    //          / \   / \
    //          768 770 198 1272
    //          / \   / \
    //          770 772 198 1276
    //          / \   / \
    //          772 774 198 1280
    //          / \   / \
    //          774 776 198 1284
    //          / \   / \
    //          776 778 198 1288
    //          / \   / \
    //          778 780 198 1292
    //          / \   / \
    //          780 782 198 1296
    //          / \   / \
    //          782 784 198 1300
    //          / \   / \
    //          784 786 198 1304
    //          / \   / \
    //          786 788 198 1308
    //          / \   / \
    //          788 790 198 1312
    //          / \   / \
    //          790 792 198 1316
    //          / \   / \
    //          792 794 198 1320
    //          / \   / \
    //          794 796 198 1324
    //          / \   / \
    //          796 798 198 1328
    //          / \   / \
    //          798 800 198 1332
    //          / \   / \
    //          800 802 198 1336
    //          / \   / \
    //          802 804 198 1340
    //          / \   / \
    //          804 806 198 1344
    //          / \   / \
    //          806 808 198 1348
    //          / \   / \
    //          808 810 198 1352
    //          / \   / \
    //          810 812 198 1356
    //          / \   / \
    //          812 814 198 1360
    //          / \   / \
    //          814 816 198 1364
    //          / \   / \
    //          816 818 198 1368
    //          / \   / \
    //          818 820 198 1372
    //          / \   / \
    //          820 822 198 1376
    //          / \   / \
    //          822 824 198 1380
    //          / \   / \
    //          824 826 198 1384
    //          / \   / \
    //          826 828 198 1388
    //          / \   / \
    //          828 830 198 1392
    //          / \   / \
    //          830 832 198 1396
    //          / \   / \
    //          832 834 198 1400
    //          / \   / \
    //          834 836 198 1404
    //          / \   / \
    //          
```

```
// 2

//ej: n10n7n6xxn9xxn15n12xxn20n8xxn14xn2xx

bintree<int> arbol(10);
arbol.insert_left(arbol.root(), 7);
arbol.insert_right(arbol.root(), 15);
arbol.insert_left(arbol.root().left(), 6);
arbol.insert_right(arbol.root().left(), 9);
arbol.insert_left(arbol.root().right(), 12);
arbol.insert_right(arbol.root().right(), 20);
arbol.insert_left(arbol.root().right().right(), 8);
arbol.insert_right(arbol.root().right().right(), 14);
arbol.insert_right(arbol.root().right().right().right(), 2);

prune_to_level(arbol,1);

cout << "Preorden del arbol resultante: ";
for (bintree<int>::preorder_iterator i = arbol.begin_preorder(); i!=arbol.end_preorder(); ++i)
    cout << *i << " ";
cout << endl;

cout << "Listado por niveles: ";
for (bintree<int>::level_iterator i = arbol.begin_level(); i!=arbol.end_level(); ++i)
    cout << *i << " ";
cout << endl;

}
```

```
#include <iostream>
#include <queue>
#include <limits>
#include "bintree.h"

using namespace std;

bintree<int>::node siguiente_inorden(bintree<int>::node n){
    //Caso base: si llegamos a un nodo nulo,
    //devolvemos el nodo nulo
    if(!n.null()){ //Caso general: el nodo no es nulo
        if(!n.right().null()){
            //Si tiene hijo derecho, buscamos
            //el descendiente más a la izquierda del hijo derecho
            n = n.right();
            while(!n.left().null())
                n = n.left();
        }
        else{
            //Si no hay hijo derecho
            //Si el nodo es hijo izquierdo, el siguiente es el padre
            //Si el nodo es hijo derecho, ascendemos mientras sigamos
            //subiendo por hijos derechos
            while(!n.parent().null() && n.parent().right() == n)
                n = n.parent();
            //En cualquier caso, tenemos que subir al padre
            n = n.parent();
        }
    }
    //El caso base es que n sea el nodo nulo,
    //lo que correspondería al else del primer if.
    //No lo incluimos porque no es necesario hacer nada:
    //si el nodo es nulo, devolvemos null

    //Devolvemos el nodo correspondiente en cada caso
    return n;
}
```

```
int main(){

    //Creamos el árbol:
    //      10
    //      /   \
    //      6   15
    //      / \   / \
    //      5   8 12  20
    //      /   \
    //      7   13

    bintree<int> arbol(10);
    arbol.insert_left(arbol.root(), 6);
    arbol.insert_left(arbol.root().left(), 5);
    arbol.insert_right(arbol.root().left(), 8);
    arbol.insert_left(arbol.root().left().right(), 7);
    arbol.insert_right(arbol.root(), 15);
    arbol.insert_left(arbol.root().right(), 12);
    arbol.insert_right(arbol.root().right(), 20);
    arbol.insert_right(arbol.root().right().left(), 13);
```

```
//Para probar el algoritmo sobre este árbol,  
//nos situamos el nodo con clave menor.  
//Para ello, partimos de la raíz y  
//descendemos por la izquierda mientras sea posible  
bintree<int>::node n = arbol.root();  
  
//Probamos el algoritmo  
n = arbol.root();  
while (!n.left().null())  
    n = n.left();  
while (!n.null()){  
    cout << *n << " ";  
    n = siguiente_inorden(n);  
}  
cout << endl;  
}
```

5.a

k	8	16	12	41	10	62	27	65	13
h(k)	8	3	12	2	10	10	1	0	0
h_o(k)	9	6	2	9	11	8	6	11	3

$$h(k) = k \% 13$$

$$h_o(k) = 1 + k \% 11$$

$$h_i(k) = (h_{i-1}(k) + h_o(k)) \% 13$$

Todas se insertan a la primera excepto:

$$h_2(62) = (10 + 8) \% 13 = 5$$

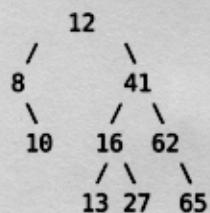
$$h_2(13) = (0 + 3) \% 13 = 3 \text{ Colisión}$$

$$h_3(13) = (3 + 3) \% 13 = 6$$

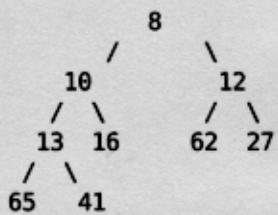
Tabla resultante

0	1	2	3	4	5	6	7	8	9	10	11	12
65	27	41	16		62	13		8		10		12

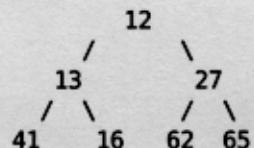
5.b Tras una rotación doble a la izquierda sobre el nodo 8 y otra simple a la izquierda sobre el nodo 16 queda el árbol:



5.c El AP0 que resulta es:



Tras hacer 2 borrados queda el AP0:



```
#include <iostream>
#include <cstdlib>
#include <list>
#include <map>

using namespace std;

bool primo(int x){
    bool es_primo = true;
    //Primos sólo pueden ser los naturales mayores que 1
    if (x<=1)
        es_primo = false;
    else if (x<=3) //2 y 3 son primos
        es_primo = true;
    else if (x%2==0) //Pares
        es_primo = false;
    else //Natural impar mayor o igual que 5
        for (int i=2; i<=x/2 && es_primo; i++)
            if (x%i==0)
                es_primo = false;
    return es_primo;
}

bool AllPrimos(const list<int> & l){
    bool todosPrimos = true;
    for (auto it = l.cbegin(); it!=l.cend() && todosPrimos; it++)
        if (!primo(*it))
            todosPrimos = false;
    return todosPrimos;
}

class Diccionario{
private:
    map <int, list<int> > datos;
public:

    list<int> & operator[](int i){
        return datos[i];
    }

    void insertar(int i, const list<int> &l){
//        datos.insert(pair<int, list<int>>(i, l));
//        datos[i].assign(l.begin(), l.end());
//        for (auto it=l.cbegin(); it!=l.cend(); it++)
//            datos[i].push_back(*it);
        datos[i] = l;
    }

    friend ostream & operator<<(ostream & os, const Diccionario & dicc){
        for(auto itd=dicc.datos.cbegin(); itd!=dicc.datos.cend(); itd++){
            os << itd->first << ": ";
            for(auto itl=itd->second.cbegin(); itl!=itd->second.cend(); itl++)
                os << *itl << " ";
            os << endl;
        }
        return os;
    }

    class iterator{
private:
    map <int, list<int> >::iterator ini, fin;;
public:
    iterator() = default;

    bool operator==(const iterator &i) const{

```

```
    return i.ini==ini;
}

bool operator!=(const iterator &i) const{
    return i.ini!=ini;
}

pair<const int,list<int> > & operator*(){
    return *ini;
}
iterator &operator++(){
    do{
        ++ini;
    }while (ini!=fin && !AllPrimos((*ini).second));
    return *this;
}

friend class Diccionario;

};

iterator begin(){
    iterator it;
    it.ini = datos.begin();
    it.fin = datos.end();
    if (it.ini!=it.fin && !AllPrimos((*(it.ini)).second))
        ++it;
    return it;
}

iterator end(){
    iterator it;
    it.ini=datos.end();
    it.fin = datos.end();
    return it;
}

int main(){
    Diccionario c;
    Diccionario::iterator it;

    srand(time(NULL));

    for (int i=1; i<=1000; i++){
        list<int> lista;
        for (int j=0; j<3; j++)
            lista.push_back(rand()%100);
        c[i] = lista;
    }

    //cout << c;

    for(auto it=c.begin(); it!=c.end(); ++it){
        auto par = *it;
        if (AllPrimos(par.second)){
            cout << par.first << ": ";
            for(auto itl=par.second.begin(); itl!=par.second.end(); ++itl)
                cout << *itl << " ";
            cout << endl;
        }
    }
}
```