

MEMORIA TRABAJO

PACIENTES C++ Evaluación Extraordinaria



Índice

Contenido

A)Nombres	3
B) Detalles y justificación de la implementación	3
B.1) Especificación concreta de la interfaz de los TAD's implementados:.....	3
B.1.1) TAD's creados	3
B.1.2) Definición de las operaciones del TAD (Nombre, argumentos y retorno).	7
B.2 Solución adoptada: descripción de las dificultades encontradas.	9
B.3) Explicación de los métodos más destacados	11
B.4 Explicación del comportamiento del programa.....	18
B.5) Bibliografía	28

A)Nombres

David Bachiller Vela

B) Detalles y justificación de la implementación

B.1) Especificación concreta de la interfaz de los TAD's implementados:

B.1.1) TAD's creados

Arboles Binarios de Búsqueda

Un árbol de búsqueda es un tipo especial de árbol binario, en el que los elementos están ordenados de la siguiente manera:

Los elementos del hijo izquierdo son todos menores o iguales que la raíz;

Los elementos del hijo derecho son todos mayores que la raíz.

ESPECIFICACIÓN: ÁRBOLES DE BÚSQUEDA

espec *ÁRBOLES_BÚSQUEDA*[ELEMENTO≤] {elem. con orden}

usa *ÁRBOLES_BINARIOS*[ELEMENTO]

parametro formal

generos *elemento*

operaciones { todas son "*op*: *elemento elemento* → *bool*" }

_ ≤ *_* *_* < *_* *_* ≥ *_* *_* > *_* *_* = *_*

var *x, y*: *elemento*

ecuaciones

{ "*_* ≤ *_*" es una relación de orden total en el TAD *elemento* }

x = *y* = (*x* ≤ *y*) ∧ (*y* ≤ *x*)

{ "*_* < *_*" y "*_* > *_*" se definen usando "*_* ≤ *_*" y "*_* = *_*" }

fparametro

ESPECIFICACIÓN: ÁRBOLES DE BÚSQUEDA (2)

operaciones

$insert : elemento\ a_bin \rightarrow a_bin$ {inserta ordenadamente}
 $está? : elemento\ a_bin \rightarrow bool$ {¿está el dato en el árbol?}

var

$x, y : elemento; i, d : a_bin$

ecuaciones

$insert(x, \Delta) = \Delta \bullet x \bullet \Delta$
 $(y \leq x) \Rightarrow insert(y, i \bullet x \bullet d) = insert(y, i) \bullet x \bullet d$
 $(y > x) \Rightarrow insert(y, i \bullet x \bullet d) = i \bullet x \bullet insert(y, d)$
 $está?(x, \Delta) = F$
 $(y < x) \Rightarrow está?(y, i \bullet x \bullet d) = está?(y, i)$
 $(y = x) \Rightarrow está?(y, i \bullet x \bullet d) = T$
 $(y > x) \Rightarrow está?(y, i \bullet x \bullet d) = está?(y, d)$

fespec

Listas Enlazadas

Una lista L es una estructura lineal caracterizada porque no tiene puntos de acceso obligatorios (sin embargo, éstos suelen ser fijos y dependientes de la construcción).

Una lista:

O bien es vacía, en cuyo caso se denomina lista vacía.

O bien puede distinguirse un elemento x, llamado cabeza, y el resto de elementos forman una lista secundaria L', que se denomina resto de la lista inicial.

ESPECIFICACIÓN: LISTAS

{Esta especificación es para la creación básica de listas}

espec LISTA[ELEMENTO]

usa BOOLEANOS

parametro formal

generos elemento

operaciones

{igualdad entre elementos}

$_eq_ : elemento\ elemento \rightarrow bool$

fparametro

generos lista

ESPECIFICACIÓN: LISTAS (2)

{Generadoras}

$[] : \rightarrow lista$ {lista vacía}

$_::_ : elemento\ lista \rightarrow lista$ {añadir por la izquierda}

{Modificadoras}

$parcial\ resto : lista \rightarrow lista$ {eliminar primero}

$parcial\ eult : lista \rightarrow lista$ {eliminar último}

{Observadoras}

$parcial\ prim : lista \rightarrow elemento$ {primero de la lista}

$parcial\ ult : lista \rightarrow elemento$ {último de la lista}

$vacía? : lista \rightarrow bool$ {ver si tiene datos}

ESPECIFICACIÓN: LISTAS (3)

var

x : elemento

l : lista

ecuaciones de definitud

Def(*prim*(*x:l*))

Def(*ult*(*x:l*))

Def(*resto*(*x:l*))

Def(*eult*(*x:l*))

ESPECIFICACIÓN: LISTAS (4)

ecuaciones

resto(*x:l*) = *l*

vacía?(*l*)=*T* \Rightarrow *eult*(*x:l*) = []

vacía?(*l*)=*F* \Rightarrow *eult*(*x:l*) = *x:eult*(*l*)

prim(*x:l*) = *x*

vacía?(*l*)=*T* \Rightarrow *ult*(*x:l*) = *x*

vacía?(*l*)=*F* \Rightarrow *ult*(*x:l*) = *ult*(*l*)

vacía?(*[]*) = *T*

vacía?(*x:l*) = *F*

fespec

LISTAS ENLAZADAS. TIPOS

tipos

nodo-lista = **reg**

valor: elemento

sig: puntero a nodo-lista {esto es lo mínimo}

freg

lista = **reg**

longitud: nat {no siempre es necesaria}

primero: puntero a nodo-lista {cabecera de lista}

freg

ftipos

COLAS

Una cola C es una estructura lineal caracterizada porque las inserciones solo se permiten en uno de los extremos de la cola, llamado final o último, y las consultas o eliminaciones solo se permiten en el opuesto, llamado principio o primero. La cola puede no tener nada, situación que se denomina cola vacía.

Las pilas se conocen también como estructuras FIFO (First In, First Out), por el modo en que se acceden los elementos.

ESPECIFICACIÓN: COLAS

{Como no sabemos qué tipo de elementos van a formar la cola, se pone una especificación genérica y usamos un parámetro formal}

espec COLA[ELEMENTO]

usa BOOLEANOS

parametro formal

generos elemento

fparametro

generos cola

ESPECIFICACIÓN: COLAS (2)

operaciones

{crear una cola vacía}
cvacia: \rightarrow cola

Generadoras

{poner un elemento en la cola}
añadir: elemento cola \rightarrow cola

{quitar un elemento de la cola}
parcial eliminar: cola \rightarrow cola

Modificadoras

{ver el principio de la cola}
parcial primero: cola \rightarrow elemento
{ver si la cola está vacía}
vacía?: cola \rightarrow bool

Observadoras

ESPECIFICACIÓN: COLAS (3)

var c: cola; x: elemento

{Como el TAD tiene operaciones parciales hay que empezar por definir los datos sobre los que pueden usarse}

{Primera opción: indicando qué forma tienen que tener los datos}

ecuaciones de definitud

Def(eliminar(añadir(x, c)))

Def(primer(añadir(x, c)))

{Segunda opción: con las propiedades que tienen que cumplir los datos para poder usar la operación} ecuaciones de definitud

vacía?(c) = F \Rightarrow Def(eliminar(c))

vacía?(c) = F \Rightarrow Def(primero(c))

Pilas

Una pila P es una estructura lineal tal que las inserciones, las consultas y las eliminaciones solo se permiten en un único punto.

La pila puede no tener nada, situación que se denomina pila vacía. Las pilas son estructuras denominadas LIFO (Last In, First Out), nombre que hace referencia al modo en que se accede a los elementos.

ESPECIFICACIÓN: PILAS

{Como no sabemos de qué va a ser la pila, ponemos una especificación genérica y usamos un parámetro formal}

espec PILA[ELEMENTO]

usa BOOLEANOS

parametro formal

generos elemento

fparametro

generos pila

ESPECIFICACIÓN: PILAS (2)

operaciones

<i>{crear una pila vacía}</i> <i>pvacia: → pila</i> <i>{poner un elemento en la pila}</i> <i>apilar: elemento pila → pila</i>	Generadoras
<i>{quitar un elemento de la pila}</i> parcial <i>desapilar: pila → pila</i>	Modificadoras
<i>{observar la cima de la pila}</i> parcial <i>cima: pila → elemento</i> <i>{para ver si la pila está vacía}</i> <i>vacía?: pila → bool</i>	Observadoras

ESPECIFICACIÓN: PILAS (3)

var *p*: pila; *x*: elemento

{Como hay operaciones parciales hay que definir cuándo pueden usarse, es decir, sobre qué datos se aplican}

{Primera forma: utilizando las generadoras del tipo}

ecuaciones de definitud

```
Def( desapilar(apilar(x,p)) )  
Def( cima(apilar(x,p)) )
```

{Segunda forma: utilizando propiedades de los datos}

ecuaciones de definitud

```
vacía?(p) = F ⇒ Def( desapilar(p) )  
vacía?(p) = F ⇒ Def( cima(p) )
```

ESPECIFICACIÓN: PILAS (4)

{Ahora que ya sabemos cuándo puede usarse una operación vamos a ver cómo se usa. Para ello ponemos los datos como si se hubiesen obtenido mediante las generadoras (cuando sea posible)}

ecuaciones

```
desapilar( apilar(x,p) ) = p  
cima( apilar(x,p) ) = x  
vacía?( pvacia ) = T  
vacía?( apilar(x,p) ) = F
```

fespec

B.1.2) Definición de las operaciones del TAD (Nombre, argumentos y retorno).

La estructura de colocación será: **nombre**(argumentos)->retorno

De árboles:

insertar (nodoArbol abb, Paciente* p)->árbol

insertar (Paciente* p)-> (insertar(raíz,p))

este insertar usa la operación anterior

dibujarNodo (vector<string>& output, vector<string>& linkAbove, pnodoAbb nodo, int nivel, int p, char linkChar)-> árbol dibujado

altura (nodaArbol abb)-> int

dibujar ()->árbol dibujado

mostrarPostOrdenArbol ()-> (postOrden(raíz)) usa siguiente operación

postOrden (nodoArbol abb)-> información de los pacientes

eliminarPreordeArbol ()->(eliminarPreorden(raíz)) usa siguiente operación

eliminarPreornden (nodoArbol abb)-> árbol vacío y cola auxiliar con los pacientes

extraerCola ()-> paciente auxiliar

maximo (nodoArbol abb)-> nodoArbol

minimo (nodoArbol abb)-> nodoArbol

mostrarMenoresMayoresArbol ()-> COUTs con información de pacientes

getLongitud ()-> int

nPares ()->(nPares(raíz)) usa siguiente operación

nPares (nodoArbol abb)-> int

mostrarHojas ()->(mostrarHojas(raíz)) usa siguiente operación

mostrarHojas (nodoArbol abb)-> info pacientes que son hoja

eliminarId (int nId, nodoArbol abb)->abb y borra nodo

eliminarPorId (int nhab)->(eliminarId(nhab,raíz), int) usa método anterior

borrarNodo (nodoArbol abb)-> nodo aux y borra nodo

getCola ()->cola

De Colas:

Insertar (paciente p)-> nuevo nodo y se suma 1 a longitud

Mostrar ()-> muestra info pacientes

Eliminar ()-> elimina nodo y resta 1 a longitud

getLongitud ()->int

De Listas:

getLongitud ()->int

primero ()->paciente

ultimo ()->paciente

mostrarMenorMayorLista ()-> COUTs info pacientes

insertarOrdenado (paciente p)->lista y suma 1 a longitud

resto ()->paciente, elimina primer nodo

mostrar ()-> info pacientes

vaciarLista()->usa función resto, vacía la lista

De Pilas:

Insertar (paciente p)->nuevo nodo en pila y suma 1 a longitud

Extraer ()->nodo aux, elimina nodo y resta 1 a longitud

Mostrar ()->cout e info pacientes

getLongitud ()->int

B.2 Solución adoptada: descripción de las dificultades encontradas.

Unos cuantos de los problemas que nos han surgido son:

Insertar nodos en el árbol.

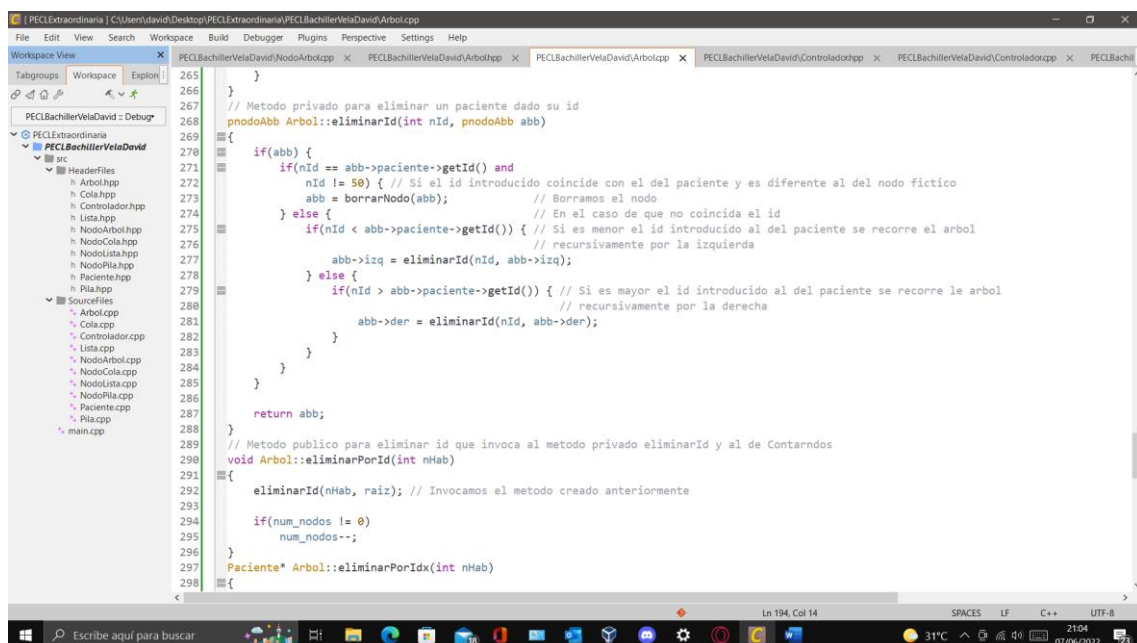
Borrar un único nodo cuando es padre, ya que se borran los hijos pero no él.

Errores de ejecución varios.

Confusiones al poner el nodo raíz en la parte privada o la pública.

Confusiones al borrar un nodo del árbol.

Problema al pasar los pacientes del árbol a la cola, nosotros tenemos un método eliminarId que va borrando nodos del árbol según el id introducido, este método es de tipo nodo y te retorna dicho nodo, luego está el método eliminarId publico para poder usarlo en el controlador que invoca este método no le pasa ningún paramtro y resta al int de numero de nodos creado 1, vale pues al ser esta función de tipo void y la de insertar cola de tipo Paciente no deja, por que la solución que se nos ocurrió es que en el método de eliminar Id, en vez de ser de tipo void que sea de tipo paciente y que te retorne dicho paciente y posteriormente poder añadirlo a la cola.



```
265 }
266 // Metodo privado para eliminar un paciente dado su id
267 pNodoArbol::eliminarId(int nId, pNodoArbol abb)
268 {
269     if (abb) {
270         if (nId == abb->paciente->getId() and
271             nId != 50) { // Si el id introducido coincide con el del paciente y es diferente al del nodo ficticio
272             abb = borrarNodo(abb); // Borramos el nodo
273         } else { // En el caso de que no coincida el id
274             if (nId < abb->paciente->getId()) { // Si es menor el id introducido al del paciente se recorre el arbol
275                 // recursivamente por la izquierda
276                 abb->izq = eliminarId(nId, abb->izq);
277             } else { // Si es mayor el id introducido al del paciente se recorre le arbol
278                 // recursivamente por la derecha
279                 abb->der = eliminarId(nId, abb->der);
280             }
281         }
282     }
283     return abb;
284 }
285 // Metodo publico para eliminar id que invoca al metodo privado eliminarId y al de ContarNodos
286 void Arbol::eliminarPorId(int nHab)
287 {
288     eliminarId(nHab, raiz); // Invocamos el metodo creado anteriormente
289     if (num_nodos != 0)
290         num_nodos--;
291 }
292 Paciente* Arbol::eliminarPorIdx(int nHab)
293 {
294 }
```

```

295 num_nodos--;
296
297 Paciente* Arbol::eliminarPorIdx(int nHab)
298 {
299     Paciente* p;
300     p = eliminarId(nHab, raiz) -> paciente; // Invocamos el metodo creado anteriormente
301     if (num_nodos != 0) {
302         num_nodos--;
303     }
304     return p;
305 }
306
307 // Metodo privado para borrar nodo
308 pnodeAbb Arbol::borrarNodo(pnodeAbb abb)
309 {
310     pnodeAbb aux;
311     Paciente* e;
312     if (abb->izq == nullptr) {
313         aux = abb->der;
314         abb->der = nullptr;
315         delete abb;
316     } else {
317         if (abb->der == nullptr) {
318             aux = abb->izq;
319             abb->izq = nullptr;
320             delete abb;
321         } else {
322             e = maximo(abb->izq) -> paciente;
323             abb->paciente = e;
324             abb->izq = eliminarId(e->getId(), abb->izq);
325             aux = abb;
326         }
327     }
328     return aux;
329 }
330
331 cola Arbol::getCola()
332 {
333 }

```

Luego lo que se implementa en el controlador es lo siguiente:

```

80 cout << "Escriba el paciente que quiere eliminar por su numero de habitacion: " << endl;
81 cin >> nHab;
82 this->Arb.eliminarPorIdx(nHab);
83 this->Arb.dibujar();
84 }
85 //Opcion G
86 void Controlador::eliminarArbolPreorden()
87 {
88     Paciente* aux;
89     while (PacientesArbol() > 0) {
90         for (int i = 0; i < contadorA; i++) {
91             aux = this->Arb.eliminarPorIdx(arrID[i]);
92         }
93         for (int i = 0; i < contadorA + contadorH; i++) {
94             aux = this->Arb.eliminarPorIdx(arrID[i] + 50);
95         }
96     }
97
98     if (aux->getApendicitis() == 1) {
99         this->colaA.insertar(aux);
100     } else {
101         this->colaH.insertar(aux);
102     }
103 }
104
105 /*
106 this->Arb.eliminarPreordeArbol();
107 while (this->Arb.getCola().getLongitud() > 0) {
108     aux = this->Arb.getCola().eliminar();
109 }
110 */
111

```

Con esto los pacientes si se borran del árbol pero no se insertan correctamente en el árbol, solución adoptada, recorrer el árbol en preorden y cada vez que se pase por la raíz pasar los elementos a una cola auxiliar , y ya de esa cola los pasaremos posteriormente a las pertinentes colas de Apendicitis y hernias.

Otra solución que se nos ocurrió es recorrerlo en preorden y cada vez que se obtenga la raíz eliminar el paciente con su id pero tampoco funciona.

```

163 {
164     if(abb != nullptr) { // Mientras que el nodo no este vacio
165         postOrden(abb->izq); // Primero recorremos el arbol por la izquierda
166         postOrden(abb->der); // Luego por la derecha
167         if(abb->paciente->getId() != 50) // Se comprueba que no se incluya el nodo ficticio creado en el constructor
168             abb->paciente->mostrarInformacionPaciente(); // Se muestra la raiz
169     }
170 }
171 // Metodo publico que invoca al metodo eliminar el arbol en preorden
172 void Arbol::eliminarPreordenArbol()
173 {
174     eliminarPreorden(raiz);
175 }
176 // Método privado para eliminar el arbol en preorden
177 void Arbol::eliminarPreorden(pnodoAbb abb)
178 {
179     Paciente* aux;
180     // En este caso para poder eliminar el arbol y poder pasarlo a la cola la idea va a ser pasar los pacientes
181     // recorridos en preorden a una cola auxiliar inicializada en el constructor y como atributo privado, y luego de e
182     // cola auxiliar pasarlos a sus pertinentes colas, tanto de apendicitis como de hernias
183     if(abb != nullptr) {
184         if(abb->paciente->getId() != 50) // Mientras que no sea el paciente ficticio
185             this->eliminarPorId(abb->paciente->getId()); // Insertamos la raiz en la cola
186         eliminarPreorden(abb->izq); // despues recorremos el arbol por la izquierda
187         eliminarPreorden(abb->der); // por ultimo por la derecha
188     }
189 }
190 Paciente* Arbol::extraerCola()
191 {

```

Debido a las soluciones encontradas hemos decidido hacerlo por una cola auxiliar.

B.3) Explicación de los métodos más destacados

Header Files

Paciente.hpp
 NodoArbol.hpp
 Arbol.hpp
 Controlador.hpp
 Cola.hpp
 NodoCola.hpp
 Pila.hpp
 NodoPila.hpp
 Lista.hpp
 NodoLista.hpp

Source Files

Paciente.cpp
 NodoArbol.cpp
 Arbol.cpp
 Controlador.cpp
 Cola.cpp
 NodoCola.cpp

Pila.cpp

NodoPila.cpp

Lista.cpp

NodoLista.cpp

Clases Paciente

Esta es la clase más importante de todo el proyecto, ya que trabajamos con la implementación de estos pacientes tanto en pilas, colas y listas como árboles.

Dentro de la clase paciente hay una serie de atributos privados:

Char dni[10], int ID , int nHabitacion y bool apendicitis.

En el caso de la enfermedad si es 1 , la enfermedad va a ser apendicitis y si es 0 , la enfermedad va a ser hernias.

Dentro de la clase paciente hemos creado los métodos:

void generarDNI() en la parte privada y en la pública; void mostrarPacientePila(), void mostrarInformacionPaciente(), void setHabitacion(int n), void setId(int id), bool getApendicitis(), char getDNI(), int getHabitacion(), int getId().

En el constructor de paciente ponemos this->apendicitis = rand()%2 para retornar 0 o 1 y asignar su enfermedad a cada paciente.

En generarDNI se genera aleatoriamente un DNI generando dos valores auxiliares aux iniciado a 0 y numDNI a 10^7 , un for en el que se genera un número aleatorio entero y se coge el resto de dividirlo entre 10 para que siempre esté entre 0 y 9, lo añade al array DNI y se lo suma a numDNI multiplicado por aux, por último divide aux entre 10.

En mostrarInformacionPaciente si el valor apendicitis(o enfermedad creado en el constructor) es 1, hace un cout que muestra su DNI, dice que tiene apendicitis, muestra su ID y su número de habitación, si el valor es 0 hace lo mismo pero diciendo que tiene hernias.

En setHabitacion se le asigna un valor n dado con this->nHabitacion = n.

En setId se le asigna un valor n dado con this->ID = n.

En getHabitacion se retorna el valor con return this->nHabitacion.

En getId se retorna el valor con return this->ID.

En getApendicitis se retorna el valor con return this->apendicitis.

El destructor está vacío.

Clase Controlador

Dentro de la clase controlador.hpp definimos todos los métodos que posteriormente vamos a utilizar en el main.

Los públicos son: void generar10pacientes(), void crearYdibujarArbol(), int PacientesArbol(), void mostrarPostOrden(), void mostrarMyoresMenoresArb(), void IDsPares(), void mostrarPacientesHoja(),

void eliminarPacienteId(), void eliminarArbolPreorden(), void mostrarPacienteColassApendicitis(), void mostrarPacientesColasHernias(), int PacientesColaApendicitis(), int PacientesColaHernias(), void vaciarColas(),void enlistarPacientesMenorMayor(), int pacientesListaApendicitis(), int pacientesListaHernias(), void mostrarPacientesListasApendicitis(), void mostrarPacientesListasHernias(), void mostrarMayoresMenoresListas(), void vaciarLista(), void apilarPacientes(), void mostrarPacientesPila(), int PacientesPila(), void vaciarPila(), void reiniciarPrograma().

Los privados son: Arbol Arb, Cola colaA, Cola colaH, Lista listaA, Lista listaH, Pila pila, int arrID[50], int arrNHab[100], int contadorA, int contadorH, int totalPacientes().

En el constructor iniciamos un árbol , una cola para apendicitis, una cola para hernias, una lista para apendicitis, una lista para hernias, una pila, un contador para apendicitis, un contador para hernias, un array ID y un array del número de habitaciones, todos vacíos o a cero. También se generan ID y nHabitación para apendicitis y hernias de manera aleatoria metiendo números sucesivos de 1 en 1 completando los arrays antes dichos (ej:1,2,3,4,5.....) y se randomiza acceso a una posición del array para posteriormente asignar un valor aleatorio en el rango válido.

En generar10pacientes se pone un if con condición si totalPacientes menor a 50 y un for de 10 iteraciones en el que se crea un nuevo paciente, si su apendicitis es valor 1, se le asigna el valor de ID y de habitación generados aleatoriamente desde el constructor, se le inserta en el árbol y se suma 1 al contador de apendicitis. Si el valor de apendicitis es 0 se le asigna el valor de ID y de habitación generados aleatoriamente desde el constructor, se le inserta en el árbol y se suma 1 al contador de hernias. Si el valor de pacientes es 50 o más salta un mensaje que dice que está el espacio completo.

En crearYdibujarArbol se dibuja el árbol.

En mostrarPostOrden se muestra el árbol en post orden.

En mostrarMyoresMenoresArb se muestran los nodos mayor y menor del árbol.

En IDsPares se muestra un mensaje con el número de pacientes con ID par.

En mostrarPacientesHoja se muestran los pacientes que son hoja del árbol.

En eliminarPacienteID se elimina el paciente poniendo su número de habitación, se dibuja primero el árbol, se pregunta el número y después llama a la función que elimina por ID y luego muestra otra vez el árbol ya sin ese nodo.

En eliminarArbolPreorden se tiene un paciente aux, si apendicitis de aux es 1, lo inserta en la cola de apendicitis, si es 0 lo inserta en la de hernias.

En mostrarPacienteColassApendicitis utiliza mostrar para mostrar la cola de apendicitis.

En mostrarPacienteColasHernias utiliza mostrar para mostrar la cola de hernias.

En vaciarColas elimina las dos colas.

En enlistarPacientesMenorMayor pasa los pacientes de las colas a sus respectivas listas.

En mostrarPacientesListasApendicitis muestra los pacientes de la lista apendicitis.

En mostrarPacientesListasHernias muestra los pacientes de la lista hernias.

En mostrarMayoresMenoresListas muestra el mayor y menor de cada lista.

En apilarPacientes pasa los pacientes de las listas a la pila.

En mostrarPacientesPila muestra los pacientes de la pila.

En vaciarLista vacía las listas.

En vaciarPila vacía la pila.

En reiniciarPrograma elimina los elementos del árbol, de las colas, de las listas y de la pila.

Ahora se ponen 6 métodos que son para obtener longitud de pila, colas...

En totalPacientes se muestra la suma del número de pacientes en el árbol, la cola de apendicitis y la de hernias.

El destructor está vacío.

Clase Main

Esta es la clase principal del programa a función main sirve como punto de partida para la ejecución del programa. Que va a controlar la ejecución del programa dirigiendo las llamadas a otras funciones del programa.

```
int main(int argc, char** argv){
```

```
//Aquí definimos el main
```

```
{
```

Para elegir la opción deseada utilizaremos un switch case y reutilizar los métodos creados en la clase controlador.

Clase NodoArbol

Se crea un nodo con un paciente, un puntero a nodo izquierdo y un puntero a nodo derecho.

Clase Arbol

Se crean los métodos públicos: void insertar (Paciente* p), void dibujar(), int getLongitud(), void mostrarPostOrdenArbol(), void eliminarPreordeArbol(), void mostrarMenoresMayores(), int nPares(), void mostrarHojas(), void eliminarPorId(int nHab).

Se crean los métodos privados: pnodeAbb raíz, pnodeAbb insertar(pnodeAbb, Paciente* p), void dibujarNodo (vector<string>& output, vector<string>& linkAbove, pnodeAbb nodo, int nivel, int minPos, char linkChar), int altura(pnodeAbb p), void postOrden(pnodeAbb ab), void eliminarPreorden(pnodeAbb ab), pnodeAbb maximo(pnodeAbb ab), pnodeAbb minimo(pnodeAbb ab), int nPares(pnodeAbb ab), void mostrarHojas(pnodeAbb ab), pnodeAbb borrarNodo(pnodeAbb ab), pnodeAbb eliminarId(int nHab, pnodeAbb ab), int ContarNodos(pnodeAbb ab), int num_nodos.

En el constructor se crea un nuevo paciente auxiliar con ID 50, se le hace raíz haciendolo nodo y iniciando un num_nodos a 0.

En `pnodoAbb` insertar se inserta un paciente al árbol de manera recursiva, si es menor a la izquierda y sino a la derecha, con el primer nodo se tiene en cuenta la raíz, devuelve `abb`. Este es su pseudocódigo:

```
proc insertar (e:elemento, abb:a_bin)
  si vacio?(abb) entonces abb←crea_árbol(e, nil,nil)
  si no
    si e ≤ (abb^.valor) entonces
      si vacio?(abb^.izq) entonces
        abb^.izq←crea_árbol(e, nil,nil)
      si no insertar (e, abb^.izq)
      finsi
    si no
      si vacio?(abb^.der) entonces
        abb^.der←crea_árbol(e, nil,nil)
      si no insertar (e, abb^.der)
      finsi
    finsi
  finproc
```

En `void` insertar se inserta un paciente usando el método anterior.

En `int` altura devuelve la altura del nodo dado, si no existe es 0, si no tiene hijos es 1, y sino se hace esa misma función de manera recursiva a un lado y a otro y da el máximo. Aquí su pseudocódigo:

```
func altura (ab:a_bin) dev natural
  si vacio?(ab) entonces error(Árbol vacío)
  si no
    si (vacio?(izq(a))) entonces
      si (vacio?(der(a))) entonces devolver 0
      si no devolver 1 + altura(der(ab))
      finsi
    si no
      si (vacio?(der(a))) entonces
        devolver 1 + altura(izq(ab))
      si no
        devolver 1 + max(altura(izq(ab)),altura(der(ab)))
      finsi
    finsi
  finsi
finfunc
```

En `void` dibujar se dibuja el árbol con el código dado en clase.

En `void` dibujaNodo se dibuja el nodo con el código dado en clase.

En `mostrarPostOrdenArbol` se muestra el árbol en post orden.

En postOrden (usado para mostrarPostOrdenArbol) se usa recursividad para mostrar la información de los pacientes en post orden. Aquí su pseudocódigo:

```
func postorden(ab:a_bin) dev lista
var l: lista
  si vacio(ab) entonces l←[]
  si no
    l←postorden(izq(ab))
    l←l++postorden(der(ab))
    l←raiz(ab)#l
  finsi
  devolver l
finfunc
```

En eliminarPreordenArbol se usa eliminarPreornden para eliminar el árbol en pre orden.

En eliminarPreornden se elimina el árbol de manera recursiva en pre orden, se elimina nodo, luego parte izquierda y después parte derecha.

En maximo muestra de manera recursiva el nodo máximo.

En minimo muestra de manera recursiva el nodo mínimo.

En mostrarMenoresMayores se usan los dos métodos anteriores para mostrar la información de los pacientes con mayor y menor valor de ID de apendicitis y de hernias.

En getLongitud se retorna el número de nodos.

En nPares se retorna nPares de la raíz, ósea el número de ID pares.

En el siguiente nPares se usa recursividad y dos bucles if para contar el número de IDs pares del árbol. Aquí su pseudocódigo:

```
func cuantos_pares (ab:a_bin) dev natural
var par_en_raíz: natural
  si vacio?(ab) entonces devolver 0
  si no
    si es_par?(raiz(ab)) entonces par_en_raíz ← 1
    si no par_en_raíz ← 0
    finsi
    devolver par_en_raíz + cuantos_pares(izq(ab))
    + cuantos_pares(der(ab))
  finsi
finfunc
```

En mostrarHojas se retorna mostrarHojas de la raíz, ósea el número de hojas.

En el siguiente mostrarHojas se usa recursividad y dos bucles if para mostrar la información de los pacientes que son hojas del árbol.

En eliminarId se elimina un nodo del árbol de manera recursiva y con cuatro ifs siempre que no sea la raíz, después retorna el árbol.

En eliminarPorId se elimina un nodo metiendo la ID con el método anterior y resta 1 al número de nodos.

En contarNodos se usa recursividad y un if para contar el número de nodos.

En borrarNodo se emplean un par de bucles if y elementos auxiliares para eliminar un nodo y “mover” el resto del árbol para que se quede bien hecho.

El destructor está vacío.

Clase Cola

En el constructor se crea primero y ultimo con valor NULL y longitud con valor 0.

Se crean los métodos públicos: void insertar(Paciente *p), Paciente *eliminar(), void mostrar(), int getLongitud().

Y en la parte privada dos nodos primero y último, y un entero longitud.

En insertar se crea un nodo “nuevo” y se le da el valor del nodo a insertar, se usan dos if, se inserta el nodo y aumenta longitud 1.

En mostrar se crea un nodo auxiliar y se le da el valor de primero, mientras exista primero, se muestra información del paciente y al aux le das el valor del siguientes en un bucle while hasta que no queden nodos.

En eliminar se crean un nodo y pacientes auxiliares, si no existe nodo, devuelve 0, a primero le da valor siguiente, al paciente auxiliar el valor del paciente del nodo, lo borra, después si no hay primero, le da a último valor NULL y resta 1 a longitud.

En getLongitud devuelve la longitud.

El destructor está vacío.

Clase NodoCola

Se crea un nodo con un paciente y un puntero siguiente.

Clase Pila

En el constructor se crea ultimo y se le da valor NULL y se crea longitud con valor 0.

Los métodos públicos son: void insertar(Paciente* p), Paciente* extraer(), void mostrar(), int getLongitud().

En la parte privada se refleja lo dicho sobre el constructor.

En insertar se crea un nodo auxiliar nuevo, se le añade el nodo y se suma 1 a longitud.

En extraer se crea un nodo y un paciente auxiliares, si no hay ultimo devuelve null y le da valor siguiente, resta 1 a longitud, elimina nodo y devuelve p.

En mostrar se crea un auxiliar y con un while se le va dando el valor de ultimo, se muestra la info de los pacientes y le da valor siguiente.

En getLongitud se devuelve la longitud.

Clase Nodopila

Se crea un nodo con un paciente y un puntero siguiente.

Clase Lista

Se crea en el constructor primero y ultimo con valor NULL y longitud con valor 0.

En la parte pública están los métodos: void insertarOrdenado(Paciente* p), Paciente* menorID(), Paciente* Primero(), Paciente* Ultimo(), Paciente* resto(), void mostrar(), void vaciarLista(), int getLognitud(), void mostrarMenorMayorLista().

En la privada se refleja lo dicho en el constructor.

En getLongitud devuelve la longitud. Aquí su pseudocódigo:

```
fun longitud(l:lista) dev n:nat
n ← l.longitud
ffun
```

En Primero devuelve el primer paciente de la lista.

En Ultimo devuelve el último paciente de la lista.

En mostrarMenorMayorLista muestra la información de los pacientes primero y último usando las dos funciones anteriores.

En insertarOrdenado se introduce un paciente, se crean dos nodos auxiliares y un nuevo nodo con valor de p dado, sino hay valor de primero se le da ese valor y a ultimo nuevo. Sino se usan varios if para lograr insertar el paciente de manera ordenada y se suma 1 a longitud.

En resto se crea un paciente y un nodo auxiliares y a aux le das valor primero y a primero el siguiente, si no hay aux devuelve 0 y a p le da valor, borra aux y si no existe primero da valor NULL a ultimo, resta 1 a longitud y devuelve p.

En mostrar crea un nodo auxiliar y le da valor de primero, en un while mientras exista muestra la información del paciente y le da valor de siguiente.

En vaciarLista mientras haya primero usa la función anterior resto y da valor 0 a longitud.

Clase NodoLista

En la clase NodoLista se crea un paciente, un siguiente y un anterior.

B.4 Explicación del comportamiento del programa.

El programa comienza ejecutando el main.cpp, al hacer eso se nos aparece por pantalla una lista en la que pone el número de pacientes que tenemos en la pila, los que hay en las colas

(son 2, la primera para los enfermos por apendicitis y la otra siguiente por hernias) y los que hay en cada lista, la de la sala de espera del quirófano de apendicitis y la de hernias y en el árbol.

Justo debajo de eso sale una lista con opciones para operar con pacientes del hospital, son las siguientes;

Opción A: cada vez que se use esta opción el programa generará 10 pacientes aleatorios y los almacenará en el árbol (hasta un máximo de 50). Crear y dibujar el ABB en consola.

```
C:\Users\David\Desktop\PECLetraordinaria\build\Debug\bin\PECLetraordinaria.exe
      50
     /  \
    14   98
   /  \  /  \
  12  40 64  90
 /  \ /  \ /  \ /  \
5  28 43 55 62 78 93 97
/  \ /  \ /  \ /  \ /  \
15 36 47 54 57 65 86 97
/  \ /  \ /  \ /  \ /  \
22 38 61 72 84 88
      68      87

-----
Pacientes en la pila -> 0
Pacientes en las colas:
  Sala A -> 0
  Sala B -> 0
Pacientes en las listas:
  Quirófano Apendicitis -> 0
  Quirófano hernias -> 0
Pacientes en el árbol -> 30
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el árbol en postorden.
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menos ID, mayor y menos nHab.
D. Contar el numero de pacientes almacenados en el ABB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el árbol antes y después de la eliminación de dicho paciente.
G. Esta opción extraera los pacientes del árbol en preorden y los almacenara en su correspondiente cola.
H. Esta opción mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opción mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opción borrara todos los pacientes almacenados en las colas.
K. Esta opción extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.
L. Esta opción mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opción mostrara los pacientes que hay en la lista Hernias.
N. Esta opción busca en las listas y muestra los siguientes 2 pacientes, mayor y menor nHab.
O. Esta opción transfiere los pacientes de las listas a la pila.
P. Esta opción muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial.
R. Salir.

Indique la opción deseada: 
```

Opción B: esta opción mostrará los pacientes que hay en el árbol en postorden

```
C:\Users\David\Desktop\PECLetraordinaria\build\Debug\bin\PECLetraordinaria.exe
1. paciente cuyo DNI es 67997693X tiene apendicitis, su ID es: 5 y su habitacion asignada es: 141
2. paciente cuyo DNI es 45861242X tiene apendicitis, su ID es: 12 y su habitacion asignada es: 108
3. paciente cuyo DNI es 12508193X tiene apendicitis, su ID es: 22 y su habitacion asignada es: 140
4. paciente cuyo DNI es 96982554H tiene apendicitis, su ID es: 15 y su habitacion asignada es: 112
5. paciente cuyo DNI es 3211253D tiene apendicitis, su ID es: 38 y su habitacion asignada es: 167
6. paciente cuyo DNI es 44222097Z tiene apendicitis, su ID es: 36 y su habitacion asignada es: 107
7. paciente cuyo DNI es 8672294X tiene apendicitis, su ID es: 28 y su habitacion asignada es: 114
8. paciente cuyo DNI es 16161355J tiene apendicitis, su ID es: 47 y su habitacion asignada es: 168
9. paciente cuyo DNI es 81951252K tiene apendicitis, su ID es: 43 y su habitacion asignada es: 119
10. paciente cuyo DNI es 29528039Z tiene apendicitis, su ID es: 46 y su habitacion asignada es: 172
11. paciente cuyo DNI es 80668100J tiene apendicitis, su ID es: 14 y su habitacion asignada es: 161
12. paciente cuyo DNI es 34849925A tiene apendicitis, su ID es: 48 y su habitacion asignada es: 118
13. paciente cuyo DNI es 12226407K tiene hernias, su ID es: 54 y su habitacion asignada es: 230
14. paciente cuyo DNI es 63729793K tiene hernias, su ID es: 63 y su habitacion asignada es: 223
15. paciente cuyo DNI es 75812574C tiene hernias, su ID es: 57 y su habitacion asignada es: 228
16. paciente cuyo DNI es 61821197A tiene hernias, su ID es: 55 y su habitacion asignada es: 241
17. paciente cuyo DNI es 30808276G tiene hernias, su ID es: 62 y su habitacion asignada es: 240
18. paciente cuyo DNI es 87303825A tiene hernias, su ID es: 68 y su habitacion asignada es: 253
19. paciente cuyo DNI es 86265192H tiene hernias, su ID es: 72 y su habitacion asignada es: 246
20. paciente cuyo DNI es 22828967G tiene hernias, su ID es: 65 y su habitacion asignada es: 212
21. paciente cuyo DNI es 85963428Z tiene hernias, su ID es: 84 y su habitacion asignada es: 263
22. paciente cuyo DNI es 36482934Z tiene hernias, su ID es: 87 y su habitacion asignada es: 275
23. paciente cuyo DNI es 21462915H tiene hernias, su ID es: 88 y su habitacion asignada es: 267
24. paciente cuyo DNI es 28253618G tiene hernias, su ID es: 86 y su habitacion asignada es: 287
25. paciente cuyo DNI es 13174180P tiene hernias, su ID es: 78 y su habitacion asignada es: 214
26. paciente cuyo DNI es 76558254J tiene hernias, su ID es: 67 y su habitacion asignada es: 268
27. paciente cuyo DNI es 96510990Y tiene hernias, su ID es: 93 y su habitacion asignada es: 219
28. paciente cuyo DNI es 38371873H tiene hernias, su ID es: 90 y su habitacion asignada es: 272
29. paciente cuyo DNI es 80180842H tiene hernias, su ID es: 64 y su habitacion asignada es: 261
30. paciente cuyo DNI es 83512016H tiene hernias, su ID es: 98 y su habitacion asignada es: 218

-----
Pacientes en la pila -> 0
Pacientes en las colas:
  Sala A -> 0
  Sala B -> 0
Pacientes en las listas:
  Quirófano Apendicitis -> 0
  Quirófano hernias -> 0
Pacientes en el árbol -> 30
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el árbol en postorden.
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menos ID, mayor y menos nHab.
D. Contar el numero de pacientes almacenados en el ABB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el árbol antes y después de la eliminación de dicho paciente.
G. Esta opción extraera los pacientes del árbol en preorden y los almacenara en su correspondiente cola.
H. Esta opción mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opción mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opción borrara todos los pacientes almacenados en las colas.

Indique la opción deseada: 
```

Opción C: buscar en el ABB y mostrar los siguientes 4 pacientes:

- El paciente con apendicitis cuyo ID es el menor.
- El paciente con apendicitis cuyo ID es el mayor.
- El paciente con hernias cuyo ID es el menor.

- El paciente con hernias cuyo ID es el mayor.

```
C:\Users\and\Desktop\PC\Listas y Arboles\Debug\bin\PC\Listas y Arboles\Debug.exe
Apendicitis Menor ID:
El paciente cuyo DNI es 67997693X tiene apendicitis, su ID es: 5 y su habitacion asignada es: 141
Apendicitis Mayor ID:
El paciente cuyo DNI es 34849925A tiene apendicitis, su ID es: 48 y su habitacion asignada es: 118
Hernias Menor ID:
El paciente cuyo DNI es 12265487X tiene hernias, su ID es: 54 y su habitacion asignada es: 238
Hernias Mayor ID:
El paciente cuyo DNI es 83512816F tiene hernias, su ID es: 98 y su habitacion asignada es: 218

-----
Pacientes en la pila -> 0
Pacientes en las colas:
    Sala A -> 0    Sala B -> 0
Pacientes en las listas:
    Quirofano Apendicitis -> 0    Quirofano hernias -> 0
Pacientes en el arbol -> 38
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el arbol en postorden.
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menor rehab.
D. Contar el numero de pacientes almacenados en el ABB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el arbol antes y despues de la eliminacion de dicho paciente.
G. Esta opcion mostrara todos los pacientes almacenados en la cola de apendicitis.
H. Esta opcion mostrara todos los pacientes almacenados en la cola de hernias.
I. Esta opcion borrara todos los pacientes almacenados en las colas.
J. Esta opcion extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.
L. Esta opcion mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opcion mostrara los pacientes que hay en la lista Hernias.
N. Esta opcion buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor rehab.
O. Esta opcion transfiere los pacientes de las listas a la pila.
P. Esta opcion muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial.
R. Salir.

Indique la opcion deseada: _
```

Opción D: contar el número de pacientes almacenados en el ABB cuyos IDs son pares.

```
C:\Users\and\Desktop\PC\Listas y Arboles\Debug\bin\PC\Listas y Arboles\Debug.exe
1. numero de Pacientes cuyo ID es par es: 19

-----
Pacientes en la pila -> 0
Pacientes en las colas:
    Sala A -> 0    Sala B -> 0
Pacientes en las listas:
    Quirofano Apendicitis -> 0    Quirofano hernias -> 0
Pacientes en el arbol -> 38
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el arbol en postorden.
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menor rehab.
D. Contar el numero de pacientes almacenados en el ABB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el arbol antes y despues de la eliminacion de dicho paciente.
G. Esta opcion mostrara todos los pacientes almacenados en la cola de apendicitis.
H. Esta opcion mostrara todos los pacientes almacenados en la cola de hernias.
I. Esta opcion borrara todos los pacientes almacenados en las colas.
J. Esta opcion extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.
L. Esta opcion mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opcion mostrara los pacientes que hay en la lista Hernias.
N. Esta opcion buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor rehab.
O. Esta opcion transfiere los pacientes de las listas a la pila.
P. Esta opcion muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial.
R. Salir.

Indique la opcion deseada: _
```

Opción E: mostrar los pacientes que se encuentran almacenados en un nodo hoja.


```

C:\Users\al\Desktop\PC\Extraordinariu\hubs\Debug\bin\PC\hachier\hachier.exe
1 paciente cuyo DNI es 67997691X tiene apendicitis, su ID es: 5 y su habitacion asignada es: 141
1 paciente cuyo DNI es 12508309X tiene apendicitis, su ID es: 22 y su habitacion asignada es: 146
1 paciente cuyo DNI es 3211125D tiene apendicitis, su ID es: 38 y su habitacion asignada es: 167
1 paciente cuyo DNI es 16161355J tiene apendicitis, su ID es: 47 y su habitacion asignada es: 168
1 paciente cuyo DNI es 12256407X tiene hernias, su ID es: 54 y su habitacion asignada es: 238
1 paciente cuyo DNI es 63729795G tiene hernias, su ID es: 61 y su habitacion asignada es: 223
1 paciente cuyo DNI es 87303825B tiene hernias, su ID es: 68 y su habitacion asignada es: 251
1 paciente cuyo DNI es 85963428B tiene hernias, su ID es: 84 y su habitacion asignada es: 263
1 paciente cuyo DNI es 36494346 tiene hernias, su ID es: 87 y su habitacion asignada es: 279
1 paciente cuyo DNI es 7065254I tiene hernias, su ID es: 97 y su habitacion asignada es: 268

-----
Pacientes en la pila -> 0
Pacientes en las colas:
    Solo A -> 0        Solo B -> 0
Pacientes en las listas:
    Quirofono Apendicitis -> 0    Quirofono hernias -> 0
Pacientes en el arbol -> 38
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el arbol en postorden.
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menos ID, mayor y menos rehab.
D. Contar el numero de pacientes almacenados en el ABB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el arbol antes y despues de la eliminacion de dicho paciente.
G. Esta opcion extraera los pacientes del arbol en preorden y los almacenara en su correspondiente cola..
H. Esta opcion mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opcion mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opcion borrara todos los pacientes almacenados en las colas.
K. Esta opcion extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.
L. Esta opcion mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opcion mostrara los pacientes que hay en la lista Hernias.
N. Esta opcion buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor rehab.
O. Esta opcion transfiere los pacientes de las listas a la pila.
P. Esta opcion muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial.
R. Salir.

Indique la opcion deseada:

```

Opción F: eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el árbol antes y después de la eliminación de dicho paciente.

```

C:\Users\al\Desktop\PC\Extraordinariu\hubs\Debug\bin\PC\hachier\hachier.exe

          50
        /  \
       48   98
      /  \  /  \
     14  64 64  98
    /  \ /  \ /  \ /  \
   12 48 43 55 62 98 93 97
  /  \ /  \ /  \ /  \ /  \
 5  38 43 55 62 98 93 97
 /  \ /  \ /  \ /  \ /  \
15 36 47 54 57 65 78 86 97
 /  \ /  \ /  \ /  \ /  \
22 38 61 72 84 88 87

Escriba el paciente que quiere eliminar por su numero de habitacion:
38

          50
        /  \
       48   98
      /  \  /  \
     14  64 64  98
    /  \ /  \ /  \ /  \
   12 48 43 55 62 98 93 97
  /  \ /  \ /  \ /  \ /  \
 5  38 43 55 62 98 93 97
 /  \ /  \ /  \ /  \ /  \
15 36 47 54 57 65 78 86 97
 /  \ /  \ /  \ /  \ /  \
22 38 61 72 84 88 87

Pacientes en la pila -> 0
Pacientes en las colas:
    Solo A -> 0        Solo B -> 0
Pacientes en las listas:
    Quirofono Apendicitis -> 0    Quirofono hernias -> 0
Pacientes en el arbol -> 29
-----

```

Opción G: esta opción extraerá los pacientes del árbol en preorden y los almacenará en su correspondiente cola.

```
C:\Users\david\Desktop\PC\Ultrason\hacer\hacer\Debug\hacer\hacer.exe

Pacientes en la pila -> 0
Pacientes en las salas:
    Sala A-> 11    Sala B-> 18
Pacientes en las listas:
    Quirofano Apendicitis-> 0    Quirofano hernias-> 0
Pacientes en el arbol -> 0

-----
A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el arbol en postorden
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menor n°hab.
D. Contar el numero de pacientes almacenados en el ABB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el arbol antes y despues de la eliminacion de dicho paciente.
G. Esta opcion extraera los pacientes del arbol en preorden y los almacenara en su correspondiente cola..
H. Esta opcion mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opcion mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opcion borrara todos los pacientes almacenados en las colas.
K. Esta opcion extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.
L. Esta opcion mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opcion mostrara los pacientes que hay en la lista Hernias.
N. Esta opcion buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor n°hab.
O. Esta opcion transfiere los pacientes de las listas a la pila.
P. Esta opcion muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial
R. Salir.

Indique la opcion deseada:
```

Opción H: esta opción mostrará todos los pacientes almacenados en la cola de apendicitis.

```
C:\Users\david\Desktop\PC\Ultrason\hacer\hacer\Debug\hacer\hacer.exe

1 paciente cuyo DNI es 34849925A tiene apendicitis, su ID es: 43 y su habitacion asignada es: 118
1 paciente cuyo DNI es 80681980J tiene apendicitis, su ID es: 14 y su habitacion asignada es: 161
1 paciente cuyo DNI es 15861242M tiene apendicitis, su ID es: 12 y su habitacion asignada es: 100
1 paciente cuyo DNI es 67997691X tiene apendicitis, su ID es: 5 y su habitacion asignada es: 141
1 paciente cuyo DNI es 32113252O tiene apendicitis, su ID es: 18 y su habitacion asignada es: 167
1 paciente cuyo DNI es 86772294X tiene apendicitis, su ID es: 28 y su habitacion asignada es: 114
1 paciente cuyo DNI es 96982554H tiene apendicitis, su ID es: 15 y su habitacion asignada es: 112
1 paciente cuyo DNI es 125081936 tiene apendicitis, su ID es: 22 y su habitacion asignada es: 146
1 paciente cuyo DNI es 446291807 tiene apendicitis, su ID es: 36 y su habitacion asignada es: 107
1 paciente cuyo DNI es 81953252C tiene apendicitis, su ID es: 43 y su habitacion asignada es: 119
1 paciente cuyo DNI es 16163355J tiene apendicitis, su ID es: 47 y su habitacion asignada es: 168

-----
Pacientes en la pila -> 0
Pacientes en las salas:
    Sala A-> 11    Sala B-> 18
Pacientes en las listas:
    Quirofano Apendicitis-> 0    Quirofano hernias-> 0
Pacientes en el arbol -> 0

-----
A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el arbol en postorden
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menor n°hab.
D. Contar el numero de pacientes almacenados en el ABB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el arbol antes y despues de la eliminacion de dicho paciente.
G. Esta opcion extraera los pacientes del arbol en preorden y los almacenara en su correspondiente cola..
H. Esta opcion mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opcion mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opcion borrara todos los pacientes almacenados en las colas.
K. Esta opcion extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.
L. Esta opcion mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opcion mostrara los pacientes que hay en la lista Hernias.
N. Esta opcion buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor n°hab.
O. Esta opcion transfiere los pacientes de las listas a la pila.
P. Esta opcion muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial
R. Salir.

Indique la opcion deseada: 1
```

Opción I: esta opción mostrará todos los pacientes almacenados en la cola de hernias.

```
C:\herramientas\Desktop\PC\Uso de estructuras de datos\Debug\hola\PC\hola.exe
El paciente cuyo DNI es 835128181 tiene hernias, su ID es: 98 y su habitacion asignada es: 218
El paciente cuyo DNI es 0618988181 tiene hernias, su ID es: 64 y su habitacion asignada es: 265
El paciente cuyo DNI es 398888776 tiene hernias, su ID es: 62 y su habitacion asignada es: 208
El paciente cuyo DNI es 61821197A tiene hernias, su ID es: 55 y su habitacion asignada es: 241
El paciente cuyo DNI es 12226607X tiene hernias, su ID es: 54 y su habitacion asignada es: 216
El paciente cuyo DNI es 7281274C tiene hernias, su ID es: 57 y su habitacion asignada es: 228
El paciente cuyo DNI es 01729753G tiene hernias, su ID es: 61 y su habitacion asignada es: 223
El paciente cuyo DNI es 38371873H tiene hernias, su ID es: 90 y su habitacion asignada es: 272
El paciente cuyo DNI es 13131388F tiene hernias, su ID es: 78 y su habitacion asignada es: 214
El paciente cuyo DNI es 22828807G tiene hernias, su ID es: 65 y su habitacion asignada es: 212
El paciente cuyo DNI es 86265392H tiene hernias, su ID es: 72 y su habitacion asignada es: 246
El paciente cuyo DNI es 87383825B tiene hernias, su ID es: 68 y su habitacion asignada es: 251
El paciente cuyo DNI es 28233818G tiene hernias, su ID es: 86 y su habitacion asignada es: 287
El paciente cuyo DNI es 05963828B tiene hernias, su ID es: 84 y su habitacion asignada es: 263
El paciente cuyo DNI es 21462919I tiene hernias, su ID es: 88 y su habitacion asignada es: 267
El paciente cuyo DNI es 36492934F tiene hernias, su ID es: 87 y su habitacion asignada es: 275
El paciente cuyo DNI es 96338896V tiene hernias, su ID es: 81 y su habitacion asignada es: 219
El paciente cuyo DNI es 76658254J tiene hernias, su ID es: 97 y su habitacion asignada es: 268

-----
Pacientes en la pila -> 0
Pacientes en las colas:
    Sala A -> 11    Sala B -> 18
Pacientes en las listas:
    Quirofono Apendicitis -> 0    Quirofono hernias -> 0
Pacientes en el arbol -> 0
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el arbol en postorden
C. Buscar en el ANB y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menor rehab.
D. Contar el numero de pacientes almacenados en el ANB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el arbol antes y despues de la eliminacion de dicho paciente.
G. Esta opcion extraera los pacientes del arbol en preorden y los almacenara en su correspondiente cola..
H. Esta opcion mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opcion mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opcion borrara todos los pacientes almacenados en las colas.
K. Esta opcion extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.
L. Esta opcion mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opcion mostrara los pacientes que hay en la lista Hernias.
N. Esta opcion buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor rehab.
O. Esta opcion transfiera los pacientes de las listas a la pila.
P. Esta opcion muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial
R. Salir.

Indique la opcion deseada: 
```

Opción J: esta opción borrará todos los pacientes almacenados en las colas.

```
C:\herramientas\Desktop\PC\Uso de estructuras de datos\Debug\hola\PC\hola.exe
Pacientes en la pila -> 0
Pacientes en las colas:
    Sala A -> 0    Sala B -> 0
Pacientes en las listas:
    Quirofono Apendicitis -> 0    Quirofono hernias -> 0
Pacientes en el arbol -> 0
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el arbol en postorden
C. Buscar en el ANB y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menor rehab.
D. Contar el numero de pacientes almacenados en el ANB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el arbol antes y despues de la eliminacion de dicho paciente.
G. Esta opcion extraera los pacientes del arbol en preorden y los almacenara en su correspondiente cola..
H. Esta opcion mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opcion mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opcion borrara todos los pacientes almacenados en las colas.
K. Esta opcion extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.
L. Esta opcion mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opcion mostrara los pacientes que hay en la lista Hernias.
N. Esta opcion buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor rehab.
O. Esta opcion transfiera los pacientes de las listas a la pila.
P. Esta opcion muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial
R. Salir.

Indique la opcion deseada: )
```

Opción K: esta opción extraerá los pacientes de la cola y los almacenará en las listas ordenados de menor a mayor por número de habitación.

```
C:\Users\David\Desktop\PCClockheritaCienDev> java C:\Users\David\Desktop\PCClockheritaCienDev.java
```

```
-----  
Pacientes en la pila -> 0  
Pacientes en las colas:  
    Sala A-> 0      Sala B-> 0  
Pacientes en las listas:  
    Quirofano Apendicitis-> 11      Quirofano hernias-> 18  
Pacientes en el arbol -> 0  
-----  
  
A. Generar 30 pacientes de forma aleatoria y almacenarlos en el Arbol.  
B. Mostrar los pacientes que hay en el arbol en postorden.  
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menor rebab..  
D. Contar el numero de pacientes almacenados en el ABB cuyos IDs son pares.  
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.  
F. Eliminar un paciente indicando por su ID (que se pide desde consola). Mostrar el arbol antes y despues de la eliminacion de dicho paciente.  
G. Esta opcion extraera los pacientes del arbol en preorder y los almacenara en su correspondiente cola..  
H. Esta opcion mostrara todos los pacientes almacenados en la cola de apendicitis.  
I. Esta opcion mostrara todos los pacientes almacenados en la cola de hernias.  
J. Esta opcion borrara todos los pacientes almacenados en las colas.  
K. Esta opcion extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.  
L. Esta opcion mostrara los pacientes que hay en la lista Apendicitis.  
M. Esta opcion mostrara los pacientes que hay en la lista Hernias..  
N. Esta opcion buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor rebab..  
O. Esta opcion transfiriere los pacientes de las listas a la pila.  
P. Esta opción muestra los pacientes de la Pila.  
Q. Reiniciar el programa a su estado inicio  
R. Salir.
```

Indique la opcion deseada: _

Opción L: esta opción mostrará los pacientes que hay en la Lista Apendicitis.

```
C:\Users\David\Desktop>gcc -g ./main.c -o main -lstdc++ -lm
1 paciente cuyo DNI es 101013553 tiene apendicitis, su ID es: 47 y su habitacion asignada es: 168
2 paciente cuyo DNI es 121132530 tiene apendicitis, su ID es: 38 y su habitacion asignada es: 167
3 paciente cuyo DNI es 006819891 tiene apendicitis, su ID es: 14 y su habitacion asignada es: 161
4 paciente cuyo DNI es 125083936 tiene apendicitis, su ID es: 22 y su habitacion asignada es: 146
5 paciente cuyo DNI es 07997683X tiene apendicitis, su ID es: 8 y su habitacion asignada es: 145
6 paciente cuyo DNI es 81953252C tiene apendicitis, su ID es: 43 y su habitacion asignada es: 119
7 paciente cuyo DNI es 34849925A tiene apendicitis, su ID es: 48 y su habitacion asignada es: 118
8 paciente cuyo DNI es 8077229AX tiene apendicitis, su ID es: 28 y su habitacion asignada es: 114
9 paciente cuyo DNI es 90802554J tiene apendicitis, su ID es: 15 y su habitacion asignada es: 112
10 paciente cuyo DNI es 15061224H tiene apendicitis, su ID es: 12 y su habitacion asignada es: 100
11 paciente cuyo DNI es 44629387T tiene apendicitis, su ID es: 36 y su habitacion asignada es: 107

-----
Pacientes en la pila -> 0
Pacientes en las colas:
    Sala A-> 0      Sala B-> 0
Pacientes en las listas:
    Quirofano Apendicitis-> 11      Quirofano hernias-> 18
Pacientes en el arbol -> 0
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el arbol en postorden
C. Buscar en el ABV y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menos edad.
D. Contar el numero de pacientes almacenados en el ABV cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicando por su ID (que se pide desde consola). Mostrar el arbol antes y despues de la eliminacion de dicho paciente.
G. Esta opcion mostrara los pacientes del arbol en preorden y los almacenara en su correspondiente cola..
H. Esta opcion mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opcion mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opcion borrara todos los pacientes almacenados en las colas.
K. Esta opcion extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.
L. Esta opcion mostrara los pacientes que hay en la Lista Apendicitis.
M. Esta opcion mostrara los pacientes que hay en la Lista Hernias.
N. Esta opcion buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor edad.
O. Esta opcion transfiere los pacientes de las listas a la pila.
P. Esta opción muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial
R. Salir.
```

Opción M: esta opción mostrará los pacientes que hay en la Lista Hernias.


```
C:\Users\user\Desktop\PC\Entidad\entidad\Debug\bin\PC\Entidad\Entidad.exe

-----
Pacientes en la pila -> 29
Pacientes en las colas:
    Sala A -> 0    Sala B -> 0
Pacientes en las listas:
    Quirofano Apendicitis -> 0    Quirofano hernias -> 0
Pacientes en el arbol -> 0
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el arbol en postorden.
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menor rehab.
D. Contar el numero de pacientes almacenados en el ABB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el arbol antes y despues de la eliminacion de dicho paciente.
G. Esta opcion extraera los pacientes del arbol en preorden y los almacenara en su correspondiente cola..
H. Esta opcion mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opcion mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opcion borrara todos los pacientes almacenados en las colas.
K. Esta opcion extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.
L. Esta opcion mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opcion mostrara los pacientes que hay en la lista Hernias.
N. Esta opcion buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor rehab.
O. Esta opcion transfiere los pacientes de las listas a la pila.
P. Esta opcion muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial.
R. Salir.

Indique la opcion deseada:
```

Opción P: esta opción muestra los pacientes de la Pila.

```
C:\Users\user\Desktop\PC\Entidad\entidad\Debug\bin\PC\Entidad\Entidad.exe

El contenido de la pila es:
El paciente cuyo DNI es 382536186 tiene hernias, su ID es: 86 y su habitacion asignada es: 287
El paciente cuyo DNI es 398880776 tiene hernias, su ID es: 62 y su habitacion asignada es: 288
El paciente cuyo DNI es 228289870 tiene hernias, su ID es: 65 y su habitacion asignada es: 212
El paciente cuyo DNI es 103743887 tiene hernias, su ID es: 78 y su habitacion asignada es: 214
El paciente cuyo DNI es 835128184 tiene hernias, su ID es: 98 y su habitacion asignada es: 218
El paciente cuyo DNI es 965189989 tiene hernias, su ID es: 93 y su habitacion asignada es: 219
El paciente cuyo DNI es 617297536 tiene hernias, su ID es: 63 y su habitacion asignada es: 223
El paciente cuyo DNI es 708125246 tiene hernias, su ID es: 57 y su habitacion asignada es: 228
El paciente cuyo DNI es 12226407X tiene hernias, su ID es: 54 y su habitacion asignada es: 230
El paciente cuyo DNI es 61821197A tiene hernias, su ID es: 55 y su habitacion asignada es: 241
El paciente cuyo DNI es 962633529 tiene hernias, su ID es: 71 y su habitacion asignada es: 246
El paciente cuyo DNI es 873838258 tiene hernias, su ID es: 68 y su habitacion asignada es: 251
El paciente cuyo DNI es 06189841N tiene hernias, su ID es: 64 y su habitacion asignada es: 261
El paciente cuyo DNI es 059634288 tiene hernias, su ID es: 84 y su habitacion asignada es: 263
El paciente cuyo DNI es 2146291189 tiene hernias, su ID es: 88 y su habitacion asignada es: 267
El paciente cuyo DNI es 766582541 tiene hernias, su ID es: 97 y su habitacion asignada es: 268
El paciente cuyo DNI es 38371873M tiene hernias, su ID es: 90 y su habitacion asignada es: 272
El paciente cuyo DNI es 34648293M tiene hernias, su ID es: 87 y su habitacion asignada es: 275
El paciente cuyo DNI es 464293877 tiene apendicitis, su ID es: 36 y su habitacion asignada es: 107
El paciente cuyo DNI es 15861242M tiene apendicitis, su ID es: 12 y su habitacion asignada es: 108
El paciente cuyo DNI es 969825548 tiene apendicitis, su ID es: 15 y su habitacion asignada es: 112
El paciente cuyo DNI es 86722948X tiene apendicitis, su ID es: 28 y su habitacion asignada es: 114
El paciente cuyo DNI es 34849925A tiene apendicitis, su ID es: 48 y su habitacion asignada es: 118
El paciente cuyo DNI es 81953252C tiene apendicitis, su ID es: 43 y su habitacion asignada es: 119
El paciente cuyo DNI es 67997693X tiene apendicitis, su ID es: 5 y su habitacion asignada es: 141
El paciente cuyo DNI es 125083936 tiene apendicitis, su ID es: 22 y su habitacion asignada es: 146
El paciente cuyo DNI es 886818891 tiene apendicitis, su ID es: 14 y su habitacion asignada es: 161
El paciente cuyo DNI es 12111353D tiene apendicitis, su ID es: 38 y su habitacion asignada es: 167
El paciente cuyo DNI es 18161355J tiene apendicitis, su ID es: 47 y su habitacion asignada es: 168

-----
Pacientes en la pila -> 28
Pacientes en las colas:
    Sala A -> 0    Sala B -> 0
Pacientes en las listas:
    Quirofano Apendicitis -> 0    Quirofano hernias -> 0
Pacientes en el arbol -> 0
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el arbol en postorden.
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menor rehab.
D. Contar el numero de pacientes almacenados en el ABB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el arbol antes y despues de la eliminacion de dicho paciente.
G. Esta opcion extraera los pacientes del arbol en preorden y los almacenara en su correspondiente cola..
H. Esta opcion mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opcion mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opcion borrara todos los pacientes almacenados en las colas.
K. Esta opcion extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por numero de habitacion.
L. Esta opcion mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opcion mostrara los pacientes que hay en la lista Hernias.
N. Esta opcion buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor rehab.
O. Esta opcion transfiere los pacientes de las listas a la pila.
P. Esta opcion muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial.
R. Salir.

Indique la opcion deseada:
```

Opción Q: reiniciar el programa a su estado inicial.

```
C:\Users\user\Desktop\PC\Listasordena\Buss\Debug\Buss\Bachier\Bachier.exe

-----
Pacientes en la pila -> 0
Pacientes en las colas:
Sala A-> 0 Sala B-> 0
Pacientes en las listas:
Quirófano Apendicitis-> 0 Quirófano hernias-> 0
Pacientes en el árbol -> 0
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el árbol en postorden
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menor reñab.
D. Contar el número de pacientes almacenados en el ABB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el árbol antes y después de la eliminación de dicho paciente.
G. Esta opción extraera los pacientes del árbol en preorden y los almacenara en su correspondiente cola..
H. Esta opción mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opción mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opción borrara todos los pacientes almacenados en las colas.
K. Esta opción extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por número de habitación.
L. Esta opción mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opción mostrara los pacientes que hay en la lista Hernias.
N. Esta opción buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor reñab.
O. Esta opción transfiera los pacientes de las listas a la pila.
P. Esta opción muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial
R. Salir.

Indique la opción deseada: _
```

Opción R: salir del programa.

```
C:\Users\user\Desktop\PC\Listasordena\Buss\Debug\Buss\Bachier\Bachier.exe

Saliedo del programa...

-----
Pacientes en la pila -> 0
Pacientes en las colas:
Sala A-> 0 Sala B-> 0
Pacientes en las listas:
Quirófano Apendicitis-> 0 Quirófano hernias-> 0
Pacientes en el árbol -> 0
-----

A. Generar 10 pacientes de forma aleatoria y almacenarlos en el Arbol.
B. Mostrar los pacientes que hay en el árbol en postorden
C. Buscar en el ABB y mostrar los siguientes 4 pacientes, mayor y menor ID, mayor y menor reñab.
D. Contar el número de pacientes almacenados en el ABB cuyos IDs son pares.
E. Mostrar los pacientes que se encuentran almacenados en un nodo hoja.
F. Eliminar un paciente indicado por su ID (que se pide desde consola). Mostrar el árbol antes y después de la eliminación de dicho paciente.
G. Esta opción extraera los pacientes del árbol en preorden y los almacenara en su correspondiente cola..
H. Esta opción mostrara todos los pacientes almacenados en la cola de apendicitis.
I. Esta opción mostrara todos los pacientes almacenados en la cola de hernias.
J. Esta opción borrara todos los pacientes almacenados en las colas.
K. Esta opción extraera los pacientes de la cola y los almacenara en las listas ordenados de menor a mayor por número de habitación.
L. Esta opción mostrara los pacientes que hay en la lista Apendicitis.
M. Esta opción mostrara los pacientes que hay en la lista Hernias.
N. Esta opción buscara en las listas y mostrara los siguientes 2 pacientes, mayor y menor reñab.
O. Esta opción transfiera los pacientes de las listas a la pila.
P. Esta opción muestra los pacientes de la Pila.
Q. Reiniciar el programa a su estado inicial
R. Salir.

Indique la opción deseada:
```

Dependiendo de la opción que se haya elegido se hará una llamada al controlador del programa para que se vayan ejecutando las funciones respectivas y nos den o hagan lo solicitado.

El programa implementa una pila en la que se meten los pacientes dados de alta, primero se meten los pacientes de apendicitis y luego los de hernias en el orden en el que estén en las listas.

Hay 2 salas de espera que serán implementadas por colas. En la primera sala se meterá a los pacientes que sufren de apendicitis y la última a los pacientes que sufren hernias.

Habrà dos posts operatorios implementados por listas simplemente enlazadas. Los pacientes entran en la lista, y se colocan en la posición que les corresponde por prioridad (número de habitación) de menor a mayor.

Habr  tambi n un  rbol con todos los pacientes ordenados seg n su n mero de habitaci n con una ra z ficticia de valor 50 en el que en un lado se encuentran los enfermos de hernias y en el otro lado los de apendicitis.

B.5) Bibliograf a

https://www.youtube.com/watch?v=dJzLmjSJc2c&list=PLWtYZ2ejMVJIUu1rEHLCOi_oibctkl0Vh (playlist de Programaci n ATS sobre c++, de la cual nos han servido los v deo referentes a  rboles, colas, listas, pilas , insertar, buscar y eliminar elementos)

Apuntes subidos al aula virtual por la profesora.

<https://www.lawebdelprogramador.com/foros/Dev-C/index1.html> (foro de c++ en la que se nos han podido ocurrir ideas varias para solucionar alg n inconveniente)

Stack overflow para dudas varias.