Primer Parcial de Programación 3 (1/10/2014)

Instituto de Computación, Facultad de Ingeniería

- Este parcial dura **4** horas y contiene **2** carillas. El total de puntos es **40**.
- En los enunciados llamamos C^* a la extensión de C al que se agrega el operador de pasaje por referencia &, las sentencias *new*, *delete*, el uso de *cout* y *cin* y el tipo `bool` predefinido en `C++`.
- NO se puede utilizar ningún tipo de material de consulta. Salvo que se indique lo contrario puede usarse todo lo visto en el teórico, práctico y laboratorio sin demostrarlo, indicando claramente lo que se está usando.

Se requiere:

i. Numerar todas las hojas e incluir en cada una el nombre y la cédula de identidad.
ii. Utilizar las hojas de un solo lado y escribir con lápiz.
iii. Iniciar cada ejercicio en hoja nueva.
iv. Poner en la carátula la cantidad de hojas entregadas, y un índice indicando en qué hojas respondieron cada problema.

Ejercicio 1 (12 puntos)

Considere la siguiente función de inserción en un heap binario.

```
void insertHeap(Heap &h, int value){

   incSize(h);
   i = size(h);
   h[i] = value;

   while (i > 1) && (h[i] < h[i div 2]){
      swap(h[i],h[i div 2]);
      i = i div 2;
   }
}
```

a) Determine la operación básica. Justifique.
b) Explique en que condiciones se da el mejor y peor caso.
c) Calcule el costo de ejecución para el mejor y peor caso en función de la operación básica.
d) Indique si las siguientes afirmaciones son verdaderas o falsas. **Justifique** su respuesta y en caso de utilizar alguna propiedad deberá **enunciarla** correctamente sin demostrarla.

i. El tiempo de ejecución para el mejor caso es $\Omega(n)$ siendo n el tamaño del heap.
ii. El tiempo de ejecución para el mejor caso es $\Theta(n)$ siendo n el tamaño del heap.
iii. $f \in O(g) y\, g \in O(f) \Rightarrow O(f) = O(g)$

Ejercicio 2 (18 puntos)

Dado un grafo **G = (V, A)** dirigido y una recorrida (DFS o BFS) siendo **T = (V, A_T)** el grafo (bosque) generado por la recorrida se clasifican las aristas de A-A_T de la siguiente manera:

- **forward**: hacia un descendiente
- **backward**: hacia un ancestro
- **cross**: entre subárboles

(A_T : aristas usadas en la recorrida o aristas **tree**)

a) Indique si las siguientes afirmaciones son verdaderas o falsas, justifique o dé un contraejemplo.

   i. En una recorrida BFS en G las aristas forward son entre niveles adyacentes.
   ii. En una recorrida BFS en G las aristas cross son entre el mismo nivel o niveles adyacentes.
   iii. En una recorrida DFS en G las aristas cross entre dos subárboles van en el mismo sentido.
   iv. Un grafo dirigido es acíclico si y solo si no tiene aristas back.

b) Escriba un algoritmo que dado un grafo y un vértice inicial realice una recorrida DFS a partir de ese vértice y retorne las aristas **back** de dicha recorrida a medida que va realizando la misma.

   ListaArista aristasBack(Grafo g, Vertice v);

c) ¿Qué modificaciones le haría al algoritmo anterior para retornar las aristas **forward**?

Ejercicio 3 (10 puntos)

Un hotel maneja información acerca de sus habitaciones y los huéspedes que se alojaron.
De cada habitación se conoce su número y comodidades.
Una habitación se identifica por su número [0..H-1] y el hotel tiene H habitaciones.
De cada huésped se conoce su documento que lo identifica y el conjunto de estadías que tuvo (habitación, fecha de inicio y fin) y un puntaje asociado a la cantidad de días y tipo de habitación en que se ha hospedado, de manera de poder definir un ranking de todos sus huéspedes.
Sea Hotel la estructura de datos utilizada para representar la realidad anterior que soporta las siguientes operaciones:

1. void imprimirHabitacionesOcupadas(Hotel h)
   Dada h, se imprime el número de cada habitación ocupada en la fecha actual. La salida deberá estar ordenada de forma ascendente por número. Esta operación debe realizarse en O(H) peor caso. Se asume conocida la fecha actual.
2. void estadiasHuesped(Hotel h, int documento)
   Dado el documento de un huésped se imprime el listado de estadías históricas (Habitación, Huésped, Fecha inicio, Fecha fin) en orden cronológico descendente. Esta operación debe realizarse en O(e) caso promedio, siendo e la cantidad de estadías del huésped.
3. void estadiasHabitación(Hotel h, int numero)
   Dado el número de una habitación se imprime el listado de estadías históricas (Habitación, Huésped, Fecha inicio, Fecha fin) en orden cronológico descendente. Esta operación deber realizarse en O(r) peor caso, siendo r la cantidad de estadías de la habitación.
4. void altaEstadía(Hotel h, int documento, int habitación, Date inicio, Date fin)
   Esta operación da de alta una estadía para el huésped y habitación durante las fechas inicio y fin indicados. Se asume que los ingresos de estadías se harán en orden cronológico, que la fecha de inicio es la actual o anterior y que no habrá superposición de fechas. Debe realizarse en O(1) caso promedio.
5. void sumarPuntaje(Hotel h, int documento, int puntos)
   Suma *puntos* al puntaje del huésped indicado. Debe realizarse en O(log k) caso promedio, siendo k la cantidad de huéspedes recibidos históricamente.
6. void rankingHuéspedes(Hotel h)
   Imprime el ranking de huéspedes en orden descendente por puntaje, debe realizarse en O(k) peor caso.

Diseñe una estructura (haga un dibujo con los comentarios pertinentes) que permita soportar las operaciones mencionadas con los órdenes de ejecución indicados. Explique detalladamente su estructura y justifique los órdenes de ejecución para cada operación.