Apellido y Nombre: _____

Carrera: _____ DNI: _____

[Llenar con letra mayúscula de imprenta GRANDE]

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática
Algoritmos y Estructuras de Datos

Algoritmos y Estructuras de Datos.
2do parcial. [16 de Noviembre de 2017]

[Ej. 1] [Clases (mínimo 50 %)]

a) **[ab]:** Dado el siguiente resumen de la implementación de árbol binario

```
class btree{
  class cell {
    elem_t t;
    cell *right,*left;
    cell() : right(NULL), left(NULL) {}
  };
  class iterator {
  private:
    cell *ptr,*father;
    side_t side;
    iterator(cell *p,side_t side_a,cell *f_a)
    : ptr(p), side(side_a), father(f_a) { }
  public:
    iterator(const iterator_t &q) { ... }
    bool operator!=(iterator_t q) { ... }
    bool operator==(iterator_t q) { ... }
    iterator() : ptr(NULL), side(NONE),
    father(NULL) { }
    iterator left() { ... }
    iterator right() { ... }  //-->implementar
  }
  ...
  iterator splice(iterator to, iterator from); //-->implementar
  iterator insert(iterator it, elem_t e); //-->implementar
  ...
}
```

implementar los métodos `btree::splice`, `iterator::right` y `btree::insert`.

b) **[set-vb]:** Dada la siguiente interfaz para un conjunto (set) por vector de bits que debe contener todos los números pares desde 1 a 100:

```
const int N=50;
typedef char elem_t;
int index(elem_t t);
elem_t element(int j);
typedef int iterator_t;
typedef pair<iterator_t,bool> pair_t;

class set {
private:
    vector<bool> v;
    iterator_t next_aux(iterator_t p);
public:
    set();
    pair_t insert(elem_t x);
    elem_t retrieve(iterator_t p);
```

```
    void erase(iterator_t p);
    bool erase(elem_t x);
    iterator_t find(elem_t x);
    iterator_t begin();
    iterator_t end();
    iterator_t next(iterator_t p);
};
```

implemente los métodos: `begin`, `next` e `insert`, y las funciones `index` y `element` (si dentro de alguno de ellos utiliza otro método de la misma clase set, deberá implementarlo también).

[Ej. 2] [Operativos (mínimo 50 %)]

a) **[Huffman]** Dados los caracteres siguientes con sus correspondientes probabilidades, construir el código binario siguiendo el algoritmo de Huffman y encodar la palabra `MESSI`.
P(A)=0.25, P(M)=0.10, P(S)=0.20, P(O)=0.10, P(I)=0.05, P(L)=0.05, P(E)=0.10, P(C)=0.10, P(U)=0.05
Calcular la longitud promedio del código obtenido.

b) **[heap-sort]:** Dados los enteros `{6, 10, 5, 2, 6, 11, 9, 4, 4, 3, 12, 7}` ordenarlos por el método de *ordenamiento por montículos*. Mostrar el montículo inicial y el resultante luego de extraer cada mínimo parcial.

c) **[abb]:** Dados los enteros `{4,8,23,4,42,15,37,16}` insertarlos, en ese orden, en un *árbol binario de búsqueda*. Mostrar las operaciones necesarias para eliminar los elementos 14, 5 y 23 en ese orden.

d) **[hash]:** Insertar los enteros `{-2, 7, 5, -12, 7, 33, 12}` en una tabla de dispersión cerrada con `B=9` cubetas, con función de dispersión `h(x) = x%B` y redispersión linear. Luego eliminar el elemento `-2` e insertar el elemento `25`, en ese orden. Mostrar la tabla resultante.

[Ej. 3] [Preguntas (mínimo 60 %)]

a) Explique que es la *condición de prefijos* en codificación. ¿Cómo se representa ésta condición por medio de árboles?

b) Comente acerca del número de intercambios realizados por los algoritmos de ordenamiento lentos.

c) ¿Qué dos condiciones debe cumplir un montículo? ¿Qué nos garantiza cada una de ellas?.

d) Explique los tiempos de ejecución de quick-sort y heap-sort para el peor caso, mejor caso y caso promedio. ¿Cuando es conveniente utilizar uno u otro?

e) Explique el concepto de estabilidad de un algoritmo de ordenamiento y clasifique los algoritmos que conoce.

f) En el desarrollo de videojuegos suele ser importante resolver de forma eficiente la colisión entre dos objetos. En los juegos actuales las escenas constan de realmente muchos objetos (>10000) distribuidos en todo el espacio y para aportar realismo se debe generar cada cuadro muy rápido (mínimo 60fps). ¿Por qué no es una buena idea simplemente detectar si un elemento colisiona con cada uno de los restantes?. Proponga una estrategia basada en las estructura de datos estudiadas durante el curso para obtener una implementación más eficiente. Base su respuesta en el analisis de órdenes de ejecución.