UNIVERSIDAD ICESI	**Primer Parcial** Marzo 2 de 2009	Algoritmos y Estructuras de datos

Reglas del juego:

- Durante este examen usted no puede pedir ABSOLUTAMENTE nada prestado a sus compañeros, ni hablar con ellos.
- Durante este examen usted no podrá utilizar ningún tipo de dispositivo electrónico, entre otros, calculadoras, celulares, asistentes personales, etc.
- Su nombre y su firma a la aceptación del compromiso de no hacer fraude, en la hoja de respuestas a este examen, deben ir en lapicero. Si emplea más de una hoja márquelas TODAS de igual forma.
- Por ningún motivo puede salir del salón, antes de terminar el examen. De manera que si se retira se considerará que terminó su trabajo.
- Las respuestas deben consignarse en el cuadernillo, NO en el enunciado. Las respuestas marcadas en el enunciado no serán tenidas en cuenta.

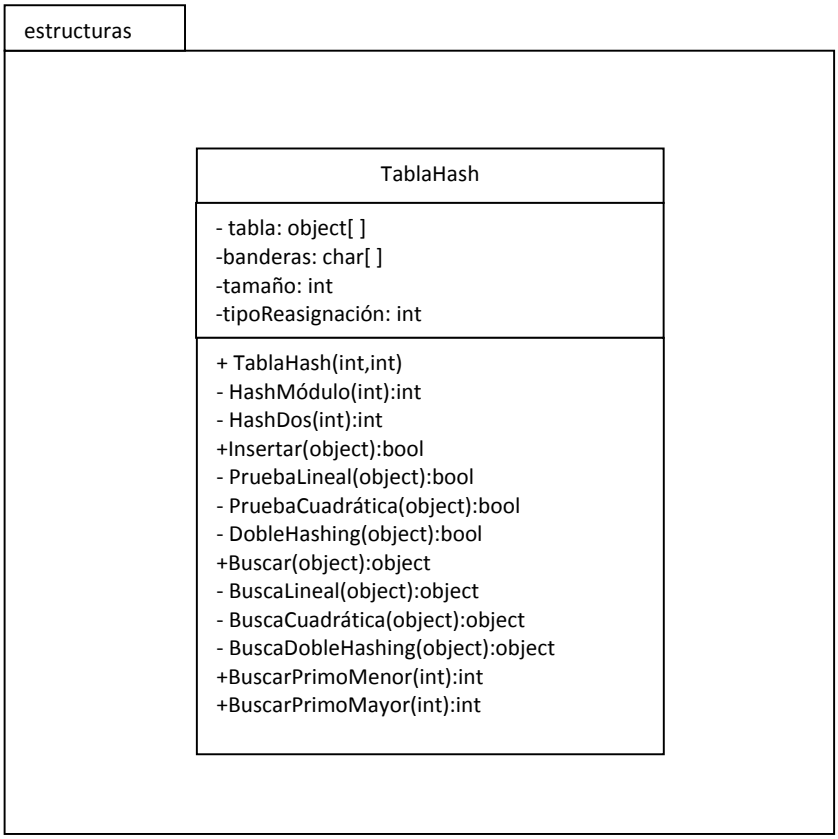
1. (1,5) Se tiene una tabla hash en la que se ingresaron los siguientes elementos en el orden especificado: 45, 12, 76, 23,17, 67, 16, 89, 10, 11.
   La tabla resultante fue:

Posición	Elemento
0	17
1	
2	
3	16
4	89
5	
6	23
7	
8	76
9	
10	10
11	45
12	12
13	
14	
15	11
16	67

   a. (0,5) Especifique qué tipo de reasignación se utilizó (debe mencionar el nombre). Las tres formas estudiadas para realizar reasignación son:
   $h_i(x) = (h(x) + i * h'(x))\%M$ con $h'(x) = q - (x\%q)$
   $h_i(x) = (h(x) + i^2)\%M$
   $h_i(x) = (h(x) + i)\%M$

   b. (1,0) Si durante el almacenamiento de la información, se acaban las casillas disponibles en una tabla hash, una forma de resolver el problema es re-calculando su tamaño. Realice el código de un método, dentro de la clase TablaHash, en donde se permita re-calcular el tamaño de la tabla, pasar todos los elementos de la tabla origen a la tabla destino, en la posición que le corresponde a cada uno de ellos. La

nueva tabla debe quedar lista para ingresar futuros elementos a partir del momento en el que se termina la ejecución del método. Para la realización de este método, debe utilizar como base el diagrama de clases que se presenta a continuación:



2. (0,5) Para el siguiente método de de búsqueda, defina si se trata de una forma secuencial o binaria y determine cuál es la complejidad temporal del mismo.

```
public static int Búsqueda(ICollection colección, IComparable clave){
            int izquierda = 0, derecha = colección.Count - 1;
            IComparable[] arreglo = new IComparable[colección.Count];
            colección.CopyTo(arreglo, 0);
            while (izquierda <= derecha)
            {
                int mitad = (izquierda + derecha) / 2;
                IComparable objetoMitad = arreglo[mitad];
                int resultado = clave.CompareTo(arreglo[mitad]);
                if (resultado == 0)
                    return mitad;
                if (resultado < 0)
                    derecha = mitad - 1;
                else
                    izquierda = mitad + 1;
            }
            return -1;
}
```

3. (1,5)
   a. (0,5) Para el siguiente método recursivo, defina: Caso base, paso recursivo, justificación de convergencia desde el paso recursivo al caso base, y qué retorna para diferentes valores de equis:

```
public static int Función(int equis)
      {
          if (equis >= 100)
          {
              return (equis - 10);
          }
          else
          {
              if (equis % 2 == 0)
              {

                  ++equis;
              }

              return (Función(Función(equis+11)));
          }
    }
```

   b. (1,0) Realice el análisis de un algoritmo recursivo que, dado un arreglo de enteros, determine si los números que contiene, tienen la forma $1^n\,0^n\,0^n\,1^n$, es decir, un cierto número *n* de unos, seguidos del doble (*2n*) de ceros, seguidos a su vez de otros *n* unos. Por ejemplo, un arreglo de tamaño 8 con las componentes [1, 1, 0, 0, 0, 0, 1, 1] cumpliría lo anterior.

4. (1,5) Para el método que se presenta a continuación, indique: (1,0) Qué método de ordenamiento es, cuál es su funcionamiento, y cuál es su complejidad temporal. (0,5) Si después de ejecutar el método anterior se ejecutara el método de búsqueda del punto dos, ¿Cuál sería la complejidad temporal total de la ejecución?

```
public static ICollection Ordenamiento(ICollection colección){
    IComparable[] arreglo = new IComparable[colección.Count];
    colección.CopyTo(arreglo, 0);

    int externo = 0;
    int interno = 0;
    IComparable temporal;

    for (externo = 1; externo < arreglo.Length; externo++)
    {
        temporal = arreglo[externo];
        interno = externo - 1;

        while ((interno >= 0) && (temporal.CompareTo(arreglo[interno]) < 0))
        {
            arreglo[interno + 1] = arreglo[interno];
            interno -= 1;
        }

        arreglo[interno + 1] = temporal;
    }
    return arreglo;
}
```