Rellenar estos cuadros:

Nombre:	
C.I.:	

Exámen 18 de diciembre de 2008

Ejercicio 1 (20 puntos)

¿Cuanto tiempo se requiere para calcular $f(x) = \sum_{i=0} a_i x^i$ en los dos siguientes casos?:

a) Usando la rutina *(1)* para realizar la exponenciación:

b) Usando la rutina *(2)* para realizar la exponenciación:

(1)

```
int  potencia (int x, int i) {
        int a=1;
        while (i > 0){
                a=a*x;
                i--;
        }
}
```

(2)

```
int  potencia (int x, int I) {
        if (i==0) return 1;
        else if (i==1) return x;
        else if (espar(i)) return potencia(x*x, n/2)
        else return potencia(x*x, n/2)*x;
 }

bool espar (int i)
{
        return ((i % 2)==0);
}
```

Ejercicio 2 (25 puntos)

Dada la siguiente especificación minima del ***TAD Array Dinámico de Enteros (ADE)***:

```
typedef struct Nodo* ADE;

// Constructoras
ADE CrearADEVacia();
// POST: devuelve  un ADE vacío.

ADE Insertar(int i, ADE l);
// POST: retorna un arreglo formado por los elementos del arreglo l, más el elemento i al
// final del mismo

// Predicados
int  cantElementos(ADE  l);
// Post: retorna la cantidad de elementos del arreglo

//Selectoras
int  obtenerElemento(ADE l, int pos);
// Pre:  l no es vacía
// Pre: 0<pos<= cantElementos(l)
// Post:Retorna  el elemento que se encuentra en la posición pos de ADE
```

ADE Resto(ADE l);
// Pre: l no es vacía
// Post: retorna el arreglo l sin su primer elemento(el elemento que se encuentra en
// la primera posición).

a) Dar una representación para el tipo ***ADE***, e implementar todas las operaciones especificadas. (*No se pueden utilizar funciones auxiliares*).

b) Utilizando el ***TAD ADE***, implemente la función ***subsecuencia*** de forma recursiva y accediendo directamente a la representación.

ADE subsecuencia(ADE l, unsigned int a, unsigned int b);
//Pre: b>=a>= cantElementos (l)>0

Que retorna la subsecuencia de l que va desde la posición 'a' inclusive, hasta la posición 'b' inclusive.
Si *b > cantElementos(l)*, se retorna el arreglo formado por los elementos desde la posición *'a'* de l hasta el final de l.

El arreglo l, no debe sufrir modificaciones dentro de la función.

Llamada	***Resultado***
subsecuencia ([1,7,-4,0,6],3,7)	[-4,0,6]
subsecuencia ([1,7,-4,0,6,4,26,100,2],2,2)	[7]
subsecuencia ([1,7,-4,0,6,4,26,100,2],2,5)	[7,-4,0,6]
subsecuencia ([1,7,-4,0,6,4,26,100,2],1,9)	[1,7,-4,0,6,4,26,100,2]

c) Sin acceder a la representación, implemente sobre el TAD ADE la operación

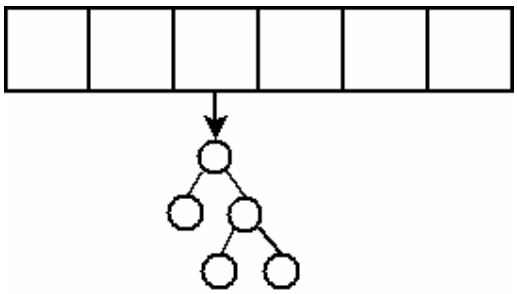
int subsecuenciaMaxima(ADE l);
//Pre : el arreglo no es vacío y por lo menos contiene un elemento mayor o igual a cero.
//Pre: no hay elementos repetidos

Que devuelve el resultado de sumar los elementos la subsecuencia máxima que se encuentra en el arreglo l.

Llamada	***Resultado***
SubsecuenciaMaxima([-1,-9,**0**])	0
SubsecuenciaMaxima([-1,-9,**0,9,2**,-7])	11
SubsecuenciaMaxima([-1,-9,**0,9,2,-1,3,-2,37,-24,38**])	62
SubsecuenciaMaxima([10,5,7,-25,**25,13,82**])	120
SubsecuenciaMaxima ([1,7,-4,0,6,4,26,100,2])	142

Comente brevemente como implemento la solución.
No se pueden utilizar funciones auxiliares.

Ejercicio 3 (20 puntos)



Dada la siguiente estructura de ***hash***, como muestra la figura, donde en cada cubeta se encuentra un árbol

a) ¿De que tipo de estrategia de resolución de colisiones se trata?

b) Si el árbol es un árbol binario de búsqueda, ¿Qué tiempos se obtienen para las operaciones de ***crearTabla***, ***insertarElemento***, ***borrarElemento*** y ***perteneceElemento***?

c) Discutir las ventajas, desventajas de esta implementación, frente a la clásica, dónde en cada cubeta se encuentra una lista simple.

Ejercicio 4 (20 puntos)

Suponga que tiene que ordenar un arreglo de enteros. Describa 2 técnicas que se puedan usar (nombre, cómo funcionan, tiempos de ejecución, ventajas y desventajas de las mismas). ***No se pide escribir el código de los algoritmos sino explicar con palabras su funcionamiento.***

Ejercicio 5 (15 puntos)

Los ***números de Catalán*** son una secuencia de números naturales que aparecen en varios problemas de conteo, como por ejemplo calcular la altura promedio de un árbol binario de búsqueda de n elementos, entre otros. Obtienen su nombre del matemático de origen belga *Eugene Charles Catalán (1814–1894).*

Se pueden calcular de la siguiente manera:

$$C_n = \begin{cases} \text{si } n = 0 & \Rightarrow 1 \\ \text{si } n > 0 & \Rightarrow \dfrac{2(2n-1)}{n+1} C_{n-1} \end{cases}$$

Se pide implementar una función recursiva que calcule el ***número de Catalán*** para un ***n*** cualquiera, use el siguiente cabezal:

```
int catalan (int n);
// Calcula el número de Catalán correspondiente a n
```

Ejercicio 6 (20 puntos)

a) De una especificación minima del TAD Árbol binario de búsqueda de naturales ABBN.

b) De una representación del TAD ABBN e implemente la operación destructora

ABBN elimNat(ABBN a, int n);
//Pre: el elemento n se encuentra en el árbol a.
//Post: Retorna el árbol a sin el elemento n

Nota: Si utiliza funciones auxiliares, deben ser implementadas.

c) Implemente de forma iterativa el procedimiento preOrden

void preOrden(ABBN a);

Que imprime en pantalla el recorrido en pre-orden del árbol a.
No se pueden utilizar funciones auxiliares.
Se pueden utilizar TADS auxiliares, en caso de hacerlo debe de dar una especificación minima de cada TAD utilizado.

****NOTA: Los ejercicios 3 y 6 son optativos entre sí, lo que significa que NO debe realizarlos a ambos, sino optar por hacer solamente uno de ellos.***