Rellenar estos cuadros: (**1 punto**)

Nombre:	¿Viene de UTU o de Secundaria?
C.I.:	

TECNÓLOGO EN INFORMÁTICA

ESTRUCTURAS DE DATOS Y ALGORITMOS

Primer Parcial

23 de Mayo de 2008

(Son 3 carillas)
Tiempo Total: 2:30 hs.

Ejercicio 1 (7 puntos)

Una progresión aritmética es una sucesión infinita de números Naturales de diferencia *d*, por ejemplo:

La sucesión: **1, 3, 5, 7,…** tiene diferencia **d = 2**.

Defina e implemente en C la función **recursiva**: *progresion_Aritmetica* descrita anteriormente, que reciba como parámetro un entero positivo *d* (para la diferencia), y un parámetro entero positivo MAX que indica el máximo entero hasta el cual llegue la sucesión (no tiene por que ser un número que pertenezca a la sucesión), e imprima en pantalla todos los números de la sucesión desde **1** (con diferencia *d*) hasta **MAX.**

Nota:

- Si MAX **no** pertenece a la sucesión **no** lo imprima.
- Si le sirve, la función puede retornar un entero.
- **No** puede usar sentencias iterativas (como son: for, while).
- **No** puede aparecer una "," al final de la lista. Fíjese en los ejemplos a continuación.

Ejemplos:

Parámetros	Llamada de función	Resultado en Pantalla
d = 2, MAX = 10	progresion_Aritmetica(2,10)	1, 3, 5, 7, 9
d = 3, MAX = 10	progresion_Aritmetica(3,10)	1, 4, 7, 10

Sugerencia:

- Antes de implementar, primero vea como es la fórmula de la progresión aritmética en términos del anterior y el siguiente.

Ejercicio 2 (16 puntos)

a) (3 ptos)
De una representación para listas de naturales (LNat). Solamente la representación y **no** las definiciones de funciones del tipo LNat.

b) (13 ptos)
Utilizando la representación de listas que dio en la parte a), implemente de forma **iterativa** una función llamada *InvertMult2* que recibe como parámetro una lista de Naturales y devuelva una lista nueva (que no comparte memoria con la lista que se pasa como parámetro), en que los elementos estén ubicados **en forma inversa** a la lista original **y el valor** que contiene cada nodo de la nueva lista es igual al valor de los nodos de la lista original **multiplicado por dos**.

Ejemplos:

Lista original	**Lista resultado**
[1, 2, 3, 5]	[10, 6, 4, 2]
[]	[]
[2]	[4]

Notas:

- Acceda directamente a la representación que declaró en la parte a)
- No se pueden usar funciones auxiliares
- La lista original se puede recorrer solamente una vez.

Ejercicio 3 16 puntos

Dados los siguientes procedimientos sobre listas de Naturales, implementadas como listas doblemente enlazadas.

- void Cons(int x, LNat &l);
- void IsEmpty(LNat l, bool &e);
- void getHead(LNat l, int &i);
- void Null(LNat &l);
- void Tail(LNat &l);

Implemente un procedimiento **recursivo** llamado *MaxSumLista* que reciba como parámetro dos listas de naturales y retorne el máximo entre la suma de todos los elementos de la primera lista y la suma de todos los elementos de la segunda lista.

Ejemplos:

Listas por parámetro	**Resultado**
L1 [3, 2, 7, 4] L2 []	16
L1 [3, 2, 200] L2 [4, 2, 4, 100]	205
L1 [3, 200] L2 [1, 2, 200]	203

Notas:

- **No** se pueden usar funciones o procedimientos auxiliares salvo los listados al principio del ejercicio.
- **No** debe modificar ninguna de las listas originales.
- **No** puede acceder a la definición de la estructura.
- **No** se deben utilizar listas auxiliares salvo (si necesita) la lista vacía.
- Cada una de las listas pasadas por parámetros solo se pueden recorrer **una** vez.
- Asuma que el máximo de la lista vacía es 0.

Soluciones:
Ejercicio 1)
Sucesión= anterior + d=siguiente

```
int progresion_Aritmetica(int d, int max){
if(max < (1 + d)){
printf("%d",1);
return 1;
}
else{
int aux = progresion_Aritmetica(d, max - d);
printf(",%d",aux + d);
return aux + d;
}
}
int main()
{
progresion_Aritmetica(2,10);
printf("\n");
progresion_Aritmetica(3,10);
printf("\n");
progresion_Aritmetica(2,1);
printf("\n");
system("PAUSE");
return 0;
}
```

Ej2)

a)

```
typedef struct Nodo{
   int info;
   struct Nodo* sig;
}*LNat;
```

b)

```
LNat InvertMult2 ( LNat l){
        LNat nueva=NULL,aux=NULL;
        while(l!=NULL){
                aux=new (Nodo);
                aux->info=2*l->info;
```

```
            aux->sig=nueva;
            nueva=aux;
            l=l->sig;
        }//while return nueva
}
```

Ej3)

```
void MaxSumLista(LNat l, LNat p, int &max){
        int maxl=0,maxp=0,aux=0;
        LNat null;
        Null(null):
        bool emptyp,emptyl;
        IsEmpty(p, emptyp);
        IsEmpty(l, emptyl);

        if(emptyl && emptyp ) {
                max=0;
        }else{
                if(!emptyp){
                        getHead(p,maxp);
                        Tail(p);
                        MaxSumLista(null, p,aux);
                        maxp=maxp+aux;
                }
                if (!emptyl){
                        getHead(l,maxl);
                        Tail(l);
                        MaxSumLista(l, null,aux);
                        maxl=maxl + aux;
                }
                if(maxl>=maxp)
                        max=maxl;
                else
                        max=maxp;
        }//fin else
}
```