Problema 1 (15 puntos)

Considere la siguiente definición de listas de enteros de memoria dinámica:

```
typedef nodoLista* Lista
struct nodoLista{
      int dato;
      Lista sig;
}
```

a) Implemente una función iterativa ***copiarPosicionesPares*** que dada una lista de enteros "l" de tipo **Lista**, retorne una copia de "l", sin compartir memoria, que contenga los elementos que se encuentran en las posiciones pares (posiciones 2, 4, 6, 8, …) y en el mismo orden que la lista parámetro. Las posiciones en una lista no vacía inician en 1. Si la lista es vacía, el resultado deberá ser la lista vacía.
La función **copiarPosicionesPares deberá tener O(n) de tiempo de ejecución en el peor caso**, siendo n el largo de la lista "l". No utilice funciones o procedimientos auxiliares, ni estructuras de datos adicionales como arreglos/vectores.

```
Lista copiarPosicionesPares(Lista l)
```

Por ejemplo, si l=[1,60,32,-3], el resultado debería ser: [60,-3].

b) Justifique muy brevemente el cumplimiento del orden exigido en la parte a) para su implementación de **copiarPosicionesPares.**

Problema 2 (15 puntos)

Considere la siguiente definición del tipo ***ABB*** de árboles binarios de búsqueda de enteros, en memoria dinámica.

```
typedef nodoABB* ABB;
struct nodoABB{
      int dato;
      ABB izq, der;
}
```

Considere la siguiente definición de listas de enteros de memoria dinámica:

```
typedef nodoLista* Lista;
struct nodoLista{
      int dato;
      Lista sig;
}
```

a) Implemente una función recursiva (**no se podrá utilizar operaciones auxiliares**) ***camino*** que, dado un árbol "*t*" de tipo ***ABB***, y un entero "x" de tipo **int**, retorne una lista de tipo **Lista** que contenga el camino a "x" desde la raíz "t".

En la lista resultado el primer elemento debe corresponder a la raíz de "t", y el último elemento debe ser "x". Si "x" no está en "t", el resultado deberá ser la lista vacía.

```
Lista camino(ABB t, int x)
```

b) Indique el orden de tiempo de ejecución en el peor caso y en el caso promedio de la función ***camino.*** Explique muy brevemente el peor caso.

Posible solución

Problema 1

a)

```
Lista copiarPosicionesPares(Lista l) {
	Lista ppio = NULL;
	Lista ult = NULL;
	unsigned int cont = 1;
	while(l != NULL){
		if(cont%2 == 0){
			Lista nodo = new nodoLista;
			nodo->dato = l->dato;
			nodo->sig = NULL;
			if(ppio == NULL)
				ppio = nodo;
			else
				ult->sig = nodo;
			ult = nodo;
		}
		l = l->sig;
		cont++;
	}
	return ppio;
}
```

b)

```
Lista copiarPosicionesPares(Lista l) { // n es la cantidad de elementos de l
      Lista ppio = NULL;            ||
      Lista ult = NULL;             || O(1)
      unsigned int cont = 1;        ||
      while(l != NULL){                                              ||
            if(cont%2 == 0){                    ||                   ||
                  Lista nodo = new nodoLista;   ||                   ||
                  nodo->dato = l->dato;         ||                   ||
                  nodo->sig = NULL;             ||                   ||
                  if(ppio == NULL)              || O(1)              ||
                        ppio = nodo;            ||                   || O(n)
                  else                          ||                   ||
                        ult->sig = nodo;        ||                   ||
                  ult = nodo;                   ||                   ||
            }                                   ||                   ||
            l = l->sig;         || O(1)                              ||
            cont++;             ||                                   ||
      }                                                              ||
      return ppio;              || O(1)
}
```

Regla de la suma:

O(max(1,n,1)) = O(n)

Problema 2

a)

```
Lista camino(ABB t, int x){
      if(t != NULL){
            if(t->dato == x){
                  Lista l = new nodoLista;
                  l->dato = t->dato;
                  l->sig = NULL;
                  return l;
            }else if(t->dato < x){
                  Lista caminoIzq = camino(t->izq, x);
                  if(caminoIzq != NULL){
                        Lista l = new nodoLista;
                        l->dato = t->dato;
                        l->sig = caminoIzq;
                        return l;
                  }else{
                        return NULL;
                  }
            }else{
                  Lista caminoDer = camino(t->der, x);
                  if(caminoDer != NULL){
                        Lista l = new nodoLista;
                        l->dato = t->dato;
                        l->sig = caminoDer;
                        return l;
                  }else{
                        return NULL;
                  }

            }
      }
      return NULL;
}
```

b)

camino es O(n) peor caso, siendo n la cantidad de elementos/nodos del árbol. El peor caso se da si el árbol degenera en una lista, y queremos buscar el camino desde la raíz hasta el menor o mayor elemento.

camino es O(log n) caso promedio.