Estructuras de Datos y Algoritmos

Segundo Semestre 2021 - Primer Parcial - Sede Buceo - Matutino/Vespertino

- ***Ejercicio 1***

Dadas las siguientes operaciones sobre una Lista Simple Ordenada de Enteros:
(la lista está ordenada de menor a mayor y no hay elementos repetidos)

lista Null ();
// Crea la lista vacia.

lista Ins (int e, lista l);
// Inserta el elemento e ordenadamente en l.

bool IsEmpty (lista l);
// Retorna "true" si l es vacía, "false" en caso contrario.

int Head (lista l);
// Retorna el primer elemento de la lista.
// Pre: l no vacía.

lista Tail (lista l);
// Retorna la lista sin su primer elemento.
// Pre: l no vacía.

Implemente ***recursivamente*** y sin usar funciones auxiliares las siguientes funciones utilizando ***exclusivamente*** las operaciones anteriores *(sin acceder a la representación interna)*:

a)

bool par (lista l);
// Retorna true si la cantidad de elementos de l es par
// false en caso contrario.

b)

bool subLista (lista s, lista l);
// Retorna true si s es una sublista de l.
// Pre: l y s están ordenadas y no hay elementos repetidos.

Ejemplos Sublistas:

l	*s*	*¿s es sublista de l?*
1 - 2 - 3 - 4 - 5	vacía	Si
1 - 2 - 3 - 4 - 5	4 - 5 - 6	No
1 - 2 - 3 - 4 - 5	2 - 4 - 5	No
1 - 2 - 3 - 4 - 5	2 - 3 - 4	Si

- ***Ejercicio 2***

a) Se pide definir el tipo *Lista Doblemente Encadenada* de Enteros *(sólo el tipo, no las operaciones)*.

b) Con el tipo definido en ***a)*** y accediendo a la representación interna implemente una función que dada una *Lista Doblemente Encadenada de Enteros*, retorne true si es simétrica y false en caso contrario.

Ejemplos Listas Simétricas y No Simétricas:

Simétricas:	*No Simétricas:*
1 - 2 - 3 - 2 - 1	1 - 2
2	1 - 2 - 3
5 - 5	1 - 5 - 5
vacía	1 - 2 - 3 – 2

- ***Ejercicio 3***

Dada la siguiente implementación de un *algoritmo de sorting*:

```
void algoritmo (int arr[], int tope){
// Ordena arr utilizando un algoritmo de sorting.
        int aux;
        for (int i = 0; i < tope ; i++){
                for (int j = 0; j < tope - i; j++){
                        if ((j + 1) < tope){
                                if (arr[j] >= arr[j + 1]){
                                        aux = arr[j];
                                        arr[j] = arr[j + 1];
                                        arr[j + 1] = aux;
                                }
                        }
                }
        }
}
```

Indique las opciones correctas *(puede haber más de una)*:

- Es una implementación de Merge Sort.
- El algoritmo es estable.
- El algoritmo es inestable.
- El orden es $O(n^3)$.
- No se puede afirmar nada sobre la estabilidad del algoritmo.
- Es una implementación de Bubble Sort.
- El orden es $O(n^2)$.
- El orden es $O(n)$.