Departamento de Ingeniería de la Información y las Comunicaciones

Facultad de Informática
UNIVERSIDAD DE MURCIA

Examen de

Tecnología de la Programación

Evaluación Continua 2023

Grupo 3.1

Realizar un Proyecto de Programación en C mediante *Code::Blocks*, llamado *NombreApellidosDNI.cdp*, el cual contenga un fichero de código fuente llamado *NombreApellidosDNI.c* que realice las siguientes tareas:

BLOQUE I (4 puntos)

1. Construir un array de números reales de tamaño $n \geq 1$, con asignación dinámica de memoria. El tamaño `n` del array debe leerse desde teclado.
2. Asignar valores aleatorios entre `-10` y `10` a los elementos del array.
3. Imprimir en pantalla los elementos del array.
4. Implementar las siguientes funciones:

```
// requerimientos: n ≥ 1
// Devuelve el mínimo de los elementos de v
double minimo(double * v, int n)

// requerimientos: n ≥ 1
// Devuelve el máximo de los elementos de v
double maximo(double * v, int n)
```

5. Imprimir en pantalla, con 2 decimales, el mínimo y el máximo del array.
6. Liberar el array.

BLOQUE II (4 puntos)

7. Declarar una estructura de datos enlazada (lineal) de números enteros, de nombre `Estructura`.
8. Implementar la siguiente función:

```
// requerimientos: a ≤ b
// Devuelve una estructura enlazada con n ≥ 0 valores enteros
// aleatorios e_i, i = 1,...,n, tales que:
// a ≤ e_1, e_i ≤ e_{i+1}, i = 1,...,n−1, y e_n ≤ b
```

```
Estructura crear_estructura(int n, int a, int b)
```

9. Crear una estructura de tamaño `n` mediante la función `crear_estructura`, con `a=0` y `b=100` (véase ejemplo de la Figura 1 para `n=5`).

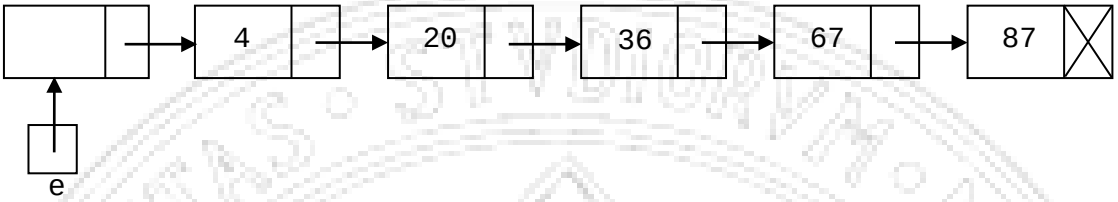

Fig. 1: Ejemplo de estructura enlazada obtenida con `e = crear_estructura(5,0,100)`.

10. Imprimir en pantalla los elementos de la estructura enlazada.
11. Liberar la estructura de datos enlazada.

BLOQUE III (4 puntos)

12. Declarar una estructura de datos enlazada para representar árboles binarios de números enteros, de nombre `ArbolBinario`.
13. Construir el árbol de la Figura 2.

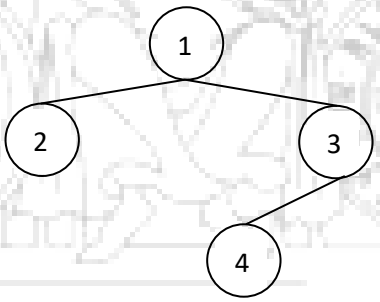

Fig. 2: Ejemplo de árbol binario.

Usar la siguiente función para crear cada nodo del árbol:

```
// Devuelve un árbol binario de un sólo nodo raíz con valor elem
// e hijos izquierdo y derecho a NULL
ArbolBinario crea_arbol(int elem)
```

14. Implementar las siguientes funciones recursivas:

```
// Imprime en pantalla los elementos del árbol binario a en
// preorden
void preorden(ArbolBinario a)

// Devuelve el producto de los elementos del árbol binario a,
// y 1 si el árbol a es vacío
int producto(ArbolBinario a)

// Libera la memoria del árbol binario a
void liberar(ArbolBinario a)
```

15. Realizar las siguientes tareas:
    15.1. Imprimir en pantalla en preorden los elementos del árbol binario con la función `preorden`.
    15.2. Imprimir en pantalla el producto de los elementos del árbol binario usando la función `producto`.
    15.3. Liberar el árbol binario `a` con la función `liberar`.

NOTA: No se requiere implementar control de errores de entrada de datos.