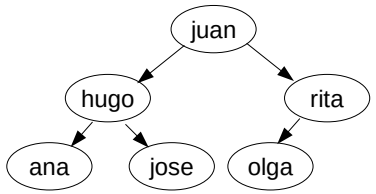
Pregunta 1 (40%)



Un conjunto se representa mediante un árbol binario de búsqueda (ABB) y se almacena en un archivo en **formato binario**. Por ejemplo, la figura de la derecha muestra un ABB y la tabla de más abajo su representación en el archivo *set.bin*. Cada fila de la tabla es un nodo del ABB y ocupa 18 bytes. La columna *pos* indica la posición del nodo en el archivo. Las columnas *izq*, *der* y *elem* son campos dentro de cada nodo. *Izq* es un valor entero de 4 bytes, en formato binario *little endian*, correspondiente a la posición del nodo raíz dela subárbol izquierdo (con respecto al inicio del archivo). Es -1 si no posee subárbol izquierdo. *Der*, también en formato binario, es la posición del nodo raíz del subárbol derecho. *Elem* contiene 10 caracteres con el nombre de un elemento del conjunto, codificado en ascii, relleno con espacios en blanco al final. No incluye terminación de string. La raíz de todo el árbol es el nodo que está al inicio del archivo. El árbol binario está ordenado lexicográficamente por el nombre del elemento y contiene al menos un nodo.

pos	**izq** *4 bytes*	**der** *4 bytes*	**elem** *10 bytes*
0	36	18	juan
18	90	-1	rita
36	54	72	hugo
54	-1	-1	ana
72	-1	-1	jose
90	-1	-1	olga

set.bin

Programe el comando *./primero.bin* que recibe como parámetro el nombre del archivo que almacena el conjunto y muestra como resultado el elemento que está primero en el orden lexicográfico, *ana* en el ejemplo. Solo debe diagnosticar error, con *perror*, si no logra abrir el archivo (*fopen* entrega NULL). Este comando se ejecutará en un procesador *little endian*. Ejemplo de uso:

```
$ ./primero.bin set.bin
ana
```

***Restricciones*:** Debe ser eficiente, lo que significa que tiene que minimizar la cantidad de nodos leídos para encontrar el elemento buscado, aprovechando que es un ABB. No puede leer secuencialmente el archivo. Debe usar *fseek* para posicionarse directamente en el desplazamiento en donde comienza cada nodo, para luego leer con *fread*.

Ayuda: Este problema es muy similar a la tarea 4 sobre archivos. Ud. debe programar la función *main*. Recorra el árbol, siempre por el subárbol izquierdo hasta llegar a un nodo sin subárbol izquierdo. El último nodo recorrido contiene el elemento solicitado.

Pregunta 2 (30%)

La función *f* de la derecha está programada en assembler Risc-V. Considere que se invoca *f* recibiendo en *a1* el entero 4 y en *a0* la dirección *d* de un arreglo de 4 enteros con los valores 3, 8, 0 y 9. La tabla de abajo muestra la ejecución de la función hasta el *sw*. Solo se anotan los saltos cuya condición es verdadera, no cuando la condición es falsa. **Prosiga llenando la tabla** con la ejecución de *addi a0,a4,4* hasta que se ejecute la instrucción *ret*. No incluya lo que ya muestra este enunciado.

```
f:
   lw   a5,0(a0)
   beq  a5,zero,L2
   ble  a1,a5,L2
   addi a0,a0,4
   j    f
L2:
   sw   a1,0(a0)
L4:
   addi a0,a0,4
   mv   a1,a5
   lw   a5,0(a0)
   sw   a1,0(a0)
   bne  a1,zero,L4
   ret
```

Instrucción	*a0*	*a1*	*a5*	*arreglo d*
	d	4		3 8 0 9
lw a5,0(a0)			3	
addi a0,a0,4	*d+4*			
j f				
lw a5,0(a0)			8	
ble a1,a5,L2				
sw a1,0(a0)				3 **4** 0 9
...				

Pregunta 3 (30%)

Traduzca la función de la derecha a assembler Risc-V.

Nota: La instrucción *break* termina el ciclo for, es decir que si *x==0*, la próxima instrucción en ejecutarse será *return b*. En un ciclo while, la condición se evalúa antes de cada iteración. En el ciclo *do ... while*, la condición se evalúa **después** de cada iteración y por lo tanto siempre se ejecuta al menos una iteración.

```
int *g(int *a) {
  int *b= a;
  for (;;) { // while(1)
    int x= *a;
    *b = x;
    if (x==0)
      break;
    do {
      a++;
    } while (x == *a );
    b++;
  }
  return b;
}
```