Pregunta 1

Parte a.- Programe la función: *int getElem(unsigned long long x, int k)*

Esta función recibe en el parámetro *x* un vector de 16 enteros sin signo de 4 bits cada uno. Si x_{63} ... x_1 x_0 son los bits de *x*, entonces el *k*-ésimo elemento del vector, con $0 \le k \le 15$, corresponde al entero representado por los bits x_{4k+3} x_{4k+2} x_{4k+1} x_{4k}. La función *getElem* entrega *el k*-ésimo elemento de *x*. Ejemplos:

```
int a= getElem(0xfb13a4f5, 0);  // a es 5
int b= getElem(0xfb13a4f5, 4);  // b es 3
```

Restricciones: No use los operadores de multiplicación, división o módulo (* / %). Use **eficientemente** los operadores de bits, sumas y restas.

Parte b.- Programe la función: *void strcat(char *s, char *r)*

Esta función agrega los carateres en el string *r* al final del string *s*. Por ejemplo:

```
char s[100] = "hola";
strcat(s, " que tal");   // s es el string "hola que tal"
```

Restricciones: No puede invocar a otras funciones dentro de su código. No use el operador de subindicación de arreglos [ ] ni su equivalente *(p+i), use aritmética de punteros.

Parte c.- Construya un circuito que dado un entero con signo de 32 bits calcule su valor absoluto. Además de la entrada y salida, utilice comparadores, restadores, multiplexores y constantes. El comparador y restador operan enteros con signo.

Pregunta 2

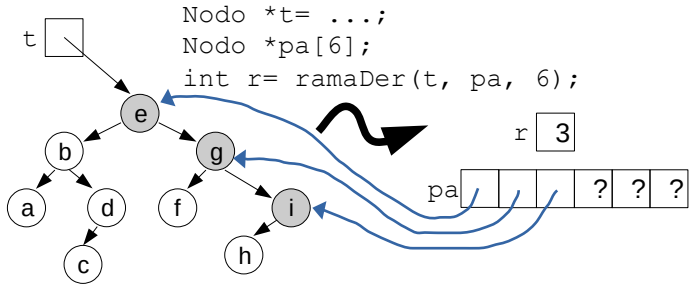
Parte i.- Programe la función: *int main(int argc, char *argv[ ])* para el comando *./linea-k*. Este comando recibe como parámetros el nombre de un archivo y un entero *k*. El comando debe desplegar en la salida estándar la *k*-ésima línea de ese archivo. Por simplicidad considere que todas las líneas tienen exactamente 80 caracteres más el término de línea *'\n'*. En el ejemplo de la derecha, el comando *./linea-k* muestra la línea 2 del archivo *lema.txt*. Preocúpese de estos errores: no logra abrir el archivo o este posee menos líneas que la solicitada.

```
$ cat lema.txt
no creo en el horoscopo
porque soy cancer
y los cancer
somos escepticos
$ ./linea-k lema.txt 2
y los cancer
$
```

Restricciones: No puede leer más de 81 caracteres del archivo. Para leer el archivo debe usar las funciones *fopen*, *fclose*, *fseek, fread* y *perror*. Opcionalmente puede usar *ftell*. No puede usar otras funciones para manipular archivos.

Parte ii.- Programe la función *ramaDer* con el encabezado del cuadro de la derecha. Esta función entrega en el arreglo *pa* hasta *n* punteros a los nodos de la rama derecha del árbol binario *a*. Debe retornar la cantidad efectiva de nodos entregados en el arreglo *pa*. En el siguiente ejemplo el árbol tiene 3 nodos en la rama derecha y por lo tanto se guardan sus direcciones en el arreglo *pa* y se retorna 3.

```
typedef struct nodo {
  char *s;
  struct nodo *izq,*der;
} Nodo;
int ramaDer(Nodo *t,
     Nodo **pa, int n);
```



Pregunta 3

A. Programe la función: *double sumarArreglo(double x[ ], int n)*

Esta función debe entregar la suma *x[0]+x[1]+...+x[n-1]*. El cálculo debe hacerse en paralelo considerando 2 cores. Para lograrlo use *fork* para crear un proceso y úselo para calcular la suma de la primera mitad del arreglo y use el proceso padre para calcular la suma de la segunda mitad.

B. El programa en assembler Risc-V de la derecha es el resultado de compilar la función *incognito*. Programe la función *equivalente* a *incognito* en C sin usar la instrucción **goto** de C. Preocúpese de *reproducir* en C todos los aspectos de la función original en assembler, en particular el valor retornado.

El encabezado de *incognito* es:

```
int *incognito(int a[], int x);
```

```
incognito:
    lw   a5,0(a0)
    beq  a1,a5,.L4
.L2:
    addi a0,a0,4
    lw   a5,0(a0)
    bne  a5,a1,.L2
.L4:
    ret
```