Pregunta 1

Programe la siguiente función: *void elimLetras(char *s)*

Esta función debe eliminar todas las letras del string *s* (caracteres a-z y A-Z). Cada letra eliminada debe ser reemplazada por un espacio en blanco al comienzo del string, de manera que se preserve el largo del string. Ejemplo:

```
char s[ ] = "ab+-43cd15AB9"; // Tiene 6 letras
elimLetras(s); // s es "      +-43159"
               // (con 6 espacios al comienzo)
```

Restricciones: Ud. no puede usar el operador de subindicación [ ], ni su equivalente **(p+i)*. Use aritmética de punteros como *p++* o *p+i*. No puede usar *malloc* ni declarar arreglos. Sí necesitará declarar punteros adicionales.

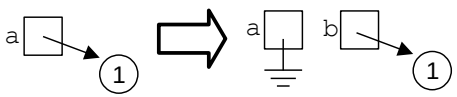
Ayuda: Use un primer puntero hacia el caracter que va a revisar y un segundo puntero hacia el lugar de destino, ambos posicionados inicialmente al final del string. Haga retroceder el primer puntero hacia el comienzo copiando caracteres en su destino. Cuando termine retroceda con el segundo puntero llenando con espacios.

Pregunta 2

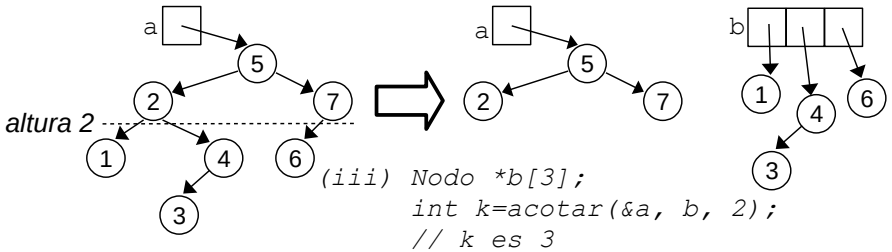
Programe la siguiente función: *int acotar(Nodo **pa, Nodo **b, int h)*

Esta función acota a *h* la altura del árbol **pa*, moviendo al arreglo *b* los subárboles no vacíos que se encuentran a una altura superior a *h*. Debe entregar la cantidad de subárboles que fueron movidos. Por simplicidad considere que el arreglo *b* es de tamaño suficiente. En los 3 siguientes ejemplos de uso el tipo de *a* es Nodo*.


```
(i) Nodo *b[1];
    int i=acotar(&a,b,5);
    // i es 0
```




```
(ii) Nodo *b[1];
     int j=acotar(&a,b,0);
     // j es 1
```




```
(iii) Nodo *b[3];
      int k=acotar(&a, b, 2);
      // k es 3
```

Ayuda: El ejemplo (i) corresponde al caso en que el árbol está vacío. El ejemplo (ii) corresponde al caso en que la altura *h* es 0. El ejemplo (iii) corresponde al caso recursivo. Una vez que haya acotado el lado izquierdo, use el valor retornado por la función *acotar* para calcular la dirección en que deben quedar en *b* los subárboles que se moverán del lado derecho.

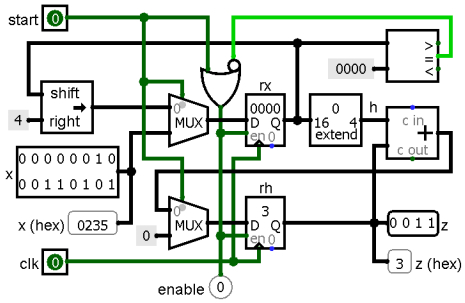
Pregunta 3

La función *pbb* del cuadro de la derecha recibe como parámetros (i) la función *gen*, que genera por ejemplo 5 cartas al azar para jugar al poker, (ii) *n*, (iii) la función *eval*, que entrega verdadero si por ejemplo las 5 cartas corresponden a 2 pares, y (iv) un puntero a una estructura con parámetros adicionales para *gen* y *eval*. La función *pbb* genera *n* juegos de cartas al azar y entrega una estimación de la probabilidad de obtener por ejemplo 2 pares. Modifique la función *pbb* de manera que se invoque *fork* para entregar la misma probabilidad pero generando *n/2* juegos en el hijo y *n-n/2* juegos en el proceso padre. No puede crear más de un proceso adicional.

```
typedef struct cards Cards;
typedef void (*GenFun)(
   void *ptr, Cards *pcards);
typedef int (*EvalFun)(
   void *ptr, Cards *pcards);
double pbb(GenFun gen, int n,
    EvalFun eval, void *ptr) {
  long long sum= 0;
  for (int i= 0; i<n; i++) {
    Cards cards;
    (*gen)(ptr, &cards);
    if ((*eval)(ptr, &cards))
      sum++;
  }
  return (double)sum / n;
}
```

Pregunta 4

I. Considere el circuito de la figura. Complete la tabla de abajo con los valores que toma cada señal hasta el ciclo 6 del reloj. El valor de la entrada *x* se mantiene constante en 0x235 (hexadecimal).



ciclo	start	rx (hex)	rh (hex)	ena-ble	h
1	0	0	3	0	0
2	**1**	0			
3	**0**	**235**			
4	0				
5	0				
6	0				

II. Traduzca la siguiente función a assembler Risc-V:

```
int pos(int *a,int x,int n){
   while (n--) {
      if (a[n]==x)
         return n;
   }
   return -1;
}
```