Pregunta 1

Parte a.- (2 puntos) Programe la función *concat_bits* declarada como:

```
typedef unsigned int uint; /* enteros sin signo */
uint concat_bits(uint x, int n, uint y, int m);
```

Si los bits de *x* son $x_{31}\ x_{30}\ \ldots\ x_1\ x_0$ en donde x_0 es el bit menos significativo y los de *y* son $y_{31}\ y_{30}\ \ldots\ y_1\ y_0$, la función concat_bits debe retornar $0\ \ldots\ 0\ x_{n-1}\ \ldots\ x_0\ y_{m-1}\ \ldots\ y_0$. Considere que *n* y *m* son menores que 32 y $n+m<=32$. Ejemplos de uso:

```
uint z1= concat_bits(0xf, 4, 0x2d, 8);     /* z1=0xf2d */
uint z2= concat_bits(0x13b, 8, 0x5b1c, 4); /* z2=0x3bc */
```

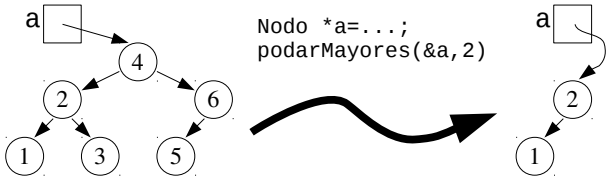
Restricciones: No puede usar los operadores de multiplicación, división o módulo. Use los operadores de bits.

Parte b.- (4 puntos) Programe la función:

```
void podarMayores(Nodo **pa, int y);
```

Esta función debe modificar el árbol de búsqueda binaria **pa* eliminando todos los nodos etiquetados con valor mayor que *y*. No necesita liberar la memoria de los nodos eliminados.

Ejemplo de uso:



Restricciones: Sea eficiente, el tiempo de ejecución debe ser proporcional a la altura del árbol en el peor caso. No puede usar ciclos (como *while* o *for*). Debe usar recursión. No puede usar *malloc*.

Pregunta 2

Parte i.- Programe la función:

```
void insertar(char *d, char *s);
```

Esta función debe insertar el string *s* al principio del string *d*. Por lo tanto el string *d* cambia, mientras que *s* no cambia. Ejemplo de uso:

```
char d[strlen("gato")+strlen("perro")+1];
        /* d es un arreglo de 10 caracteres */
strcpy(d, "perro");  /* d es "perro" */
insertar(d, "gato"); /* d es "gatoperro" */
```

Restricciones: Ud. no puede usar el operador de indexación [ ] o su equivalente **(s+i)*. Sí puede usar *s+i*, **s*, **s++*, **s--*. Es decir programe en C, no en Java. No puede pedir memoria adicional para hacer una copia de *d*.

Sea cuidadoso al desplazar el contenido original de *d* para hacer espacio para *s*. Si copia ascendentemente, el resultado final de *d* en el ejemplo será incorrectamente "*gatoperrp*". Hágalo descendentemente.

Parte ii.- Considere la siguiente función:

```
typedef void (*VoidFun)(void *ptr);
int boringEval(VoidFun fun, void *ptr) {
 (*fun)(ptr);
 return 0;
}
```

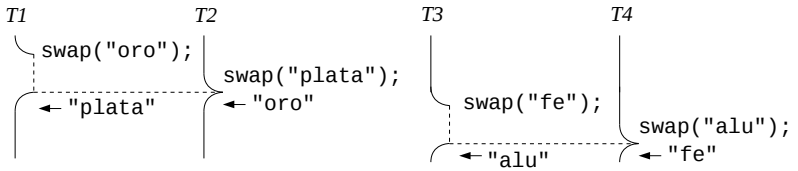
En donde *VoidFun* es el tipo de las funciones que reciben un puntero opaco (*void**) como parámetro y no retornan nada. Modifique la función *boringEval* de manera que se aborte el cálculo de *fun* si durante su ejecución el usuario se aburre y presiona *control-C* en el teclado (señal *SIGINT*). Si hay *control-C boringEval* debe retornar 1.

Pregunta 3

Programe la siguiente función usando un *mutex* y una *condición*:

```
char *swap(char *s);
```

Esta función es invocada concurrentemente desde múltiples threads y varias veces. Sirve para intercambiar strings con cualquier otro thread que invoque *swap*. La siguiente figura explica el funcionamiento de *swap*. *T1* invoca *swap* pero no hay otro thread pendiente en un *swap*, entonces *T1* queda en espera. Más tarde *T2* invoca *swap* y encuentra que *T1* está pendiente en un *swap*, entonces *T1* y *T2* intercambian los valores. Es decir en *T1 swap* retorna el string "plata" y en *T2 swap* entrega el string "oro" y continúan ejecutándose. Más tarde *T3* invoca *swap* y como no hay ningún thread pendiente, *T3* queda en espera hasta que *T4* invoca *swap* para intercambiar sus valores.



En su solución necesitará usar variables globales, como por ejemplo el mutex y la condición.