Pregunta 1

Parte a.- Programe las funciones *agregar* y *pertenecer* para manipular conjuntos de caracteres. Sus encabezados son:

```
typedef unsigned int uint;
void agregar(uint *conj, char c);
int pertenece(uint *conj, char c);
```

Estas funciones reciben un puntero *conj* que contiene la dirección de un arreglo de 8 enteros sin signo de 32 bits cada uno. Además reciben un caracter *c* que se codifica mediante un entero que puede tomar valores entre -128 y 127. En el arreglo Ud. dispone de 256 bits para indicar si cada uno de los 256 caracteres posibles está presente en el conjunto. La función *agregar* debe incluir el carácter *c* en el conjunto *conj* y la función *pertenece* debe entregar un valor distinto de 0 si el carácter *c* está presente en el conjunto y 0 si no lo está. Ejemplo de uso:

```
uint conj[8]= {0, 0, 0, 0, 0, 0, 0, 0};
int r1= pertenece(conj, 'a');  /* r1 es 0 */
agregar(conj, 'a'); agregar(conj, 'a');
int r2= pertenece(conj, '=');  /* r2 es 0 */
agregar(conj, '=');
int r3= pertenece(conj, 'a');  /* r3 es 1 */
int r4= pertenece(conj, '=');  /* r4 es 1 */
```

Restricción: en estas 2 funciones Ud. no puede usar los operadores de multiplicación, división o módulo (* / %). Use los operadores de bits.

Parte b.- Programe la siguiente función:

```
void eliminar(char *str, char *caracs);
```

Esta función elimina del string *str* todos los caracteres que aparezcan en el string *caracs*. Ejemplo de uso:

```
char *s= "los 4 puntos cardinales son 3: el norte y el sur";
char *r= malloc(strlen(s)+1);
strcpy(r, s);
eliminar(r, "aeiouaei");
printf("%s\n", r); /* ls 4 pnts crdnls sn 3: l nrt y l sr */
```

Su solución debe ser eficiente, es decir el tiempo de eliminación debe ser proporcional a la suma de los largos de *str* y *caracs*, y no al producto. Para lograrlo use la parte a.

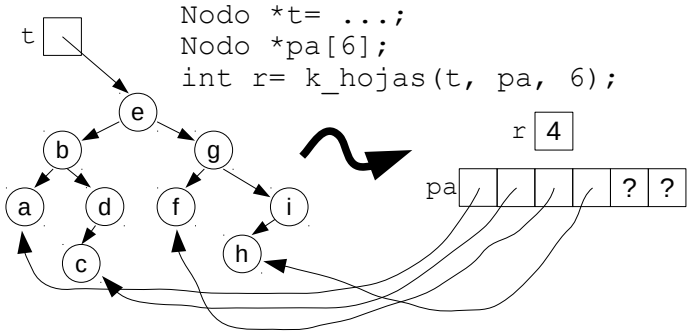
Restricción: en *eliminar* Ud. no puede usar el operador de subindicación [ ]. Use aritmética de punteros.

Pregunta 2

I. (4,5 puntos) Se define una hoja en un árbol de búsqueda binaria como un nodo que no tiene ningún hijo. Escriba la función *k_hojas* con el siguiente encabezado:

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

Esta función debe recorrer en orden el árbol *t* y guardar en el arreglo *pa* las *k* primeras hojas de *t*. Debe retornar el número efectivo de hojas que se depositaron en *pa*, que no debe exceder *k*. Ejemplo de uso:



II. (1,5 puntos) Complete en este programa los espacios marcados como … de modo que el programa no arroje errores o *warnings* durante la compilación.

```
typedef struct { … } Stru;
typedef … Fun …;
int proc(Fun f, Stru *p, char *q) {
  … r= &q;
  … s= &r;
  p->d= **s;
  return (*f)(p);
}
```