Pregunta 1

Parte a.- Programe la función *buscar* con el siguiente encabezado:

```
typedef int (*Comparador)(void *a, int k, void *x);
int buscar(void *a, int n, void *x,
           Comparador cmp);
```

La función *buscar* debe realizar una búsqueda genérica del objeto abstracto *x* en el arreglo abstracto *a* de tamaño *n*. Debe retornar el índice de *x* en el arreglo o -1 si no se encontró. La búsqueda es genérica porque no se sabe cual es el tipo concreto del arreglo *a* o del objeto *x*. Por esta razón para poder realizar las comparaciones de los elementos de *a* con *x* se recibe como parámetro *cmp*, que corresponde a una función que recibe el arreglo *a*, un índice *k* y el objeto *x,* entregando negativo, 0 o positivo según el *k*-ésimo elemento de *a* sea menor, igual o mayor que *x*.

Restricción: Ud. debe usar búsqueda binaria. El arreglo *a* está ordenado de acuerdo al orden impuesto por la función *cmp*.

A modo de ejemplo de uso de la función *buscar*, en el siguiente código la función *buscar_nombre*, busca la posición del string *nom* en el arreglo ordenado *nombres* de tamaño *n*.

```
int cmp_nombres(void *a, int k, void *x) {
  char **nombres= a, *nom= x;
  return strcmp(nombres[k], nom);
}
int buscar_nombre(char **nombres, int n, char *nom) {
  return buscar(nombres, n, nom, cmp_nombres);
}
```

Parte b.- Un archivo contiene un diccionario en el siguiente formato:

alimento:sustancia ingerida por un ser vivo:
casa:edificación construida para ser habitada:
embarcación:todo tipo de artilugio capaz de navegar sobre o bajo el agua:
lluvia:condensación del vapor de agua contenida en las nubes:
… etc ...

El primer carácter ':' separa la palabra de su definición. El segundo ':' termina la definición. Todas las líneas del archivo tienen 80 caracteres para que sea sencillo hacer acceso directo con *fseek*. Las palabras están ordenadas lexicográficamente para que se pueda buscar eficientemente en el archivo mediante búsqueda binaria. Este archivo no cabe en la memoria del computador. Programe la siguiente función:

*int consultar(char *nom_dic, char *pal, int n_lin);*

En donde *nom_dic* es el nombre del archivo, *pal* es la palabra que se busca y *n_lin* es el número de líneas del archivo (y por lo tanto el número de palabras y definiciones). Esta función entrega el número de línea en donde se encuentra *pal* en el archivo *nom_dic*. Por ejemplo si *pal* es "alimento", *consultar* debe entregar 0. Si *pal* es "lluvia", se debe entregar 3.

Restricción: Para realizar la búsqueda, Ud. debe utilizar la función *buscar* de la parte a. No reprograme nuevamente la búsqueda binaria.

A modo de recuerdo, ésta es la API para manejar archivos:

```
FILE *fopen(char *nom_arch, char *modo);   /* modo== "r" */
size_t fread(char *buf, size_t ancho_item, int n_item, FILE *file);
int fseek(FILE *file, long despl, int w);          /* w==SEEK_SET */
int fclose(FILE *file);
```

Pregunta 2

Los 5 filósofos se cansaron de comer arroz con palitos. Es muy difícil. El sabio chino accedió a colocar 3 cucharas en el centro de la mesa. Ahora un filósofo solo necesita una cuchara para comer su arroz y puede ser cualquiera de las 3 cucharas. La siguiente solución es *incorrecta*:

```
void filosofo(int i) {
  for (;;) {
    int cuchara= elegir(0, 2);
    comer(cuchara);
    pensar();
} }
```

Recuerde que hay 5 threads paralelos. Cada uno ejecuta la función *filosofo* con valores para i entre 0 y 4. La función *elegir* entrega un entero aleatorio en el rango [0,2].

Reprograme la solución de más arriba de manera que se asegure que 2 filósofos no puedan comer con la misma cuchara al mismo tiempo. Los filósofos pueden ser atendidos en cualquier orden. No necesita evitar la hambruna.

Restricción: Su solución debe ser tal que si hay un filósofo que desea comer y hay una cuchara disponible, ese filósofo debe comer con esa cuchara de inmediato. Esto implica por ejemplo que Ud. debe lograr que lleguen a comer hasta 3 filósofos en paralelo.