Parte a.- Programe en un archivo *parte-a.c* 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.- El archivo *dicc.txt* 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. El archivo contiene 12 líneas, todas de 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.

Programe en un archivo *parte-b.c* la siguiente función:

*char *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 la definición asociada a la palabra *pal* en el archivo *nom_dic* o *NULL* si no se encontró. Por ejemplo si *pal* es "alimento", *consultar* debe entregar *"sustancia ingerida por un ser vivo"*. Si *pal* es "lluvia", se debe entregar *"condensación del vapor de agua contenida en las nubes"*.

Restricciones: Para realizar la búsqueda, Ud. debe utilizar la función *buscar* de la parte a. No reprograme nuevamente la búsqueda binaria. Ud. no puede leer el archivo completo en memoria. Use *fseek*.

Recursos

Baje de material docente el archivo *t2.zip*. Contiene los archivos (i) el programa *test-parte-a.c* para verificar la parte a, (ii) el programa *test-parte-b.c* para verificar la parte b, (iii) el archivo *t2.h* con los encabezados de las funciones que Ud. debe programar y que Ud. debe incluir en los archivos *parte-a.c* y *parte-b.c*, (iv) el archivo *Makefile* para compilar su tarea, (v) el archivo *dicc.txt* con el diccionario que Ud. debe usar en la parte b, y (vi) el directorio T2-2015-1 con el enunciado y la solución de la tarea 2 del semestre otoño 2015 que es similar a esta tarea.

Compile y ejecute la parte a con el comando:

```
$ make test-parte-a
$ ./test-parte-a
```

Este comando usa las reglas incluidas en el archivo *Makefile* para compilar todos los archivos. Se le felicitará si su código funciona. Compile y ejecute la parte b con el comando:

```
$ make test-parte-b
$ ./test-parte-b
```

El programa lo felicitará si pasa todos los tests. Pero no olvide satisfacer también los requerimientos de programación enunciados más arriba.

Entrega

Ud. debe entregar mediante U-cursos un archivo `t2.zip` que incluya solo los archivos *parte-a.c* y *parte-b.c*. No incluya otros archivos por favor. No se aceptarán tareas que no pasen los tests de unidad suministrados. Se descontará medio punto por día de atraso (excepto sábados, domingos y festivos).