UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Informática I — Curso R1051 — Primer Parcial 2019

Apellido y nombre:_____ Legajo:_____

Parte Teórica

1. Dado el siguiente código indique qué se imprime en pantalla. Justifique.

```
#include <stdio.h>
#include <string.h>
int main(void) {
      char *p;
      printf("%s\n", strcpy(p, "hola"));
      return 0;
}
```

2. La variable `a` definida e inicializada según la sentencia `int *a = &b;` ocupa la dirección de memoria `0x2000` y su contenido es `0x3000`. Completar la siguiente tabla.

¿De qué tipo tiene que ser la variable b?	
Valor de c definida como `int **c = &a;`	
Resultado de `printf("%p\n", a);`	
Resultado de `printf("%x\n", *a);`	

Parte Práctica

Implemente las siguientes funciones:

1. **`int contar_palabra(char *s, char *palabra);`**
   Devuelve la cantidad de veces que aparece el string apuntado por "palabra" dentro del string apuntado por "s". Si no se cumpliese con alguna de las reglas de validación, esa aparición de "palabra" no se cuenta.

   Reglas de validación:
   1. Que "palabra" quepa en "s".
   2. "palabra" puede estar al comienzo de "s", al final, en cualquier otra parte y aparecer cero ó más veces. Pero si está antes ó después de cualquier otra palabra, incluso de otra aparición de ella misma, debe estar separada por espacio, tab, o cualquier otro signo de puntuación.

   Tips: para determinar si un caracter es espacio o tab se puede usar la función de librería `isspace()`. Para saber si es un signo de puntuación se puede usar `ispunct()`. Ambas están declaradas en `ctype.h`

Ej. supongamos la palabra “chau”, si el string apuntado por “s” es:

a. “cha” -> no cuenta
b. “chau, me voy a casa” -> cuenta
c. “chau” -> cuenta
d. “... y ahora te digo chau” -> cuenta
e “chau, chau, chau!!!” -> cuenta
f “chauchauchau?” -> no cuenta

2. **`char* elimina_palabra(char* s, char* palabra, int* cantidad_eliminada);`**
   Con las mismas reglas de validación de `contar_palabra`, `eliminar_palabra` elimina todas las apariciones de “palabra” en “s”. Devuelve un puntero al comienzo del string con todas las eliminaciones correspondientes ya realizadas, si las hubiese, para que pueda encadenarse con otras funciones. También devuelve en `cantidad_eliminadas` exactamente eso, la cantidad de veces que se eliminó la palabra. Todas las operaciones se realizarán “in-place”, o sea sobre el mismo string apuntado por “s”.

3. **`char* ordena_string(char* s, int (*fcomp)(char* a, char* b));`**
   Ordena utilizando el algoritmo bubble-sort los caracteres contenidos en el string apuntado por “s” utilizando la función apuntada por `fcomp` para realizar la comparación previa al swap. Devuelve un puntero al comienzo del string ya ordenado. Todas las operaciones se realizarán “in-place”, o sea sobre el mismo string apuntado por “s”.

   También deberá implementar dos funciones de comparación, una que permita ordenar lexicográficamente en forma ascendente y otra para realizarlo en forma descendente.

   Se adjuntan los archivos `.c`, `.h` y `Makefile` con los prototipos y esqueletos de las funciones y un `main` para poder probarlas. Además se adjuntan scripts de test para probar las validaciones.

Condiciones de aprobación

Dos de las tres funciones (`contar_palabra, eliminar_palabra, ordena_string`) bien implementadas. Sin errores de compilación, ni errores conceptuales, ni fallas por excepciones.

Tres de los cinco puntos teóricos sin fallas de conceptos y debidamente justificados, si corresponde.