Procesamiento de cadenas de caracteres y estructuras

1) Ejercicio procesamiento de cadena de caracteres:

Implementar un programa en C que lea desde la entrada estándar STDIN una línea de caracteres formada una palabra seguida de dos números enteros (“SET 5 15” por ejemplo). Luego deberá extraer las 3 partes respectivamente en un vector de char y dos variables de tipo entero. Ejemplo:

Se tiene el vector char Línea[40], cargar en la misma desde la entrada estándar “SET 5 15” y luego copiar en char cmd[10] la palabra SET y en las variables int a el 5 y en int b el 15.

Para leer la línea investigar y probar las funciones getline, getchar, scanf y fscanf de las bibliotecas stdio.h y stdlib.h

Para extraer y convertir los números se pueden utilizar las funciones atoi() o strtol()

2) Ordenar lista de cadenas mediante vector de punteros y memoria dinámica

En el archivo String_list_sort.c se presenta un programa que mediante un vector de punteros a char carga y almacena un listado de palabras ingresadas por teclado. Posteriormente se ordenan alfabéticamente y se muestra la lista ordenada en pantalla.

Se presentan dos funciones para realizar el ordenamiento mediante el método burbuja, una mediante el reordenamiento copiando las cadenas a las posiciones adecuadas del vector y el segundo que reordena los punteros almacenados en el vector de punteros sin copiar las cadenas.

Probar el comportamiento de ambas funciones, observar que para que la función que ordena los punteros sin copiar las palabras funcione se debe llamar pasándole la dirección del vector x (reorder(n,&x) en cambio la otra no lo requiere (reorder2(n,x)).

3) Tipos de almacenamiento de variables (auto y register) y parámetros del main

El archivo fib.c genera la secuencia de números de Fibonacci hasta el numero N recibido como parámetro. Utilizando 3 variables locales de la función y 1 local al for. Son variables de tipo auto.

El archivo fib2.c la función calcula el elemento N de la secuencia y en el main se muestra un mecanismo para medir el tiempo de ejecución de la función mediante funciones de la biblioteca time.h.

En el archivo fib3.c se implementa lo mismo que en el anterior y ofrece además la posibilidad de ingresar el numero N como un parámetro desde la consola al ejecutar el programa.

Con este último, medir el tiempo de ejecución del computo de diversos valores de la serie y generar una gráfica de N vs Tiempo de ejecución. Finalmente, modificar la declaración de las variables a b y c en la función para que sean de tipo register int y repetir el análisis para observar si hay algún cambio.

4) Ejemplos de estructuras:

a. Dada la definición de las siguientes estructuras:

```
#define SIZE 100
/* Información sobre la celda */
struct informacion_celda
{
    char nombre[SIZE]; /* Nombre de la celda */
    unsigned int identificador; /* Número identificador */
    float calidad_senal;              /* Calidad de la señal (entre 0 y
100) */
    struct informacion_operador * ptr_operador; /* Puntero a una segunda
estructura */
};

/* Información sobre el operador */
struct informacion_operador
{
    char nombre[SIZE]; /* Cadena de texto con el nombre */
    unsigned int prioridad;     /* Prioridad de conexión */
    unsigned int ultima_comprob;  /* Fecha de la última comprobación */
};
```

i. ¿Qué tamaño en bytes ocupa una variable de tipo struct informacion_celda en memoria?

ii. Las siguientes dos líneas declaran dos variables. ¿Cuál de ellas ocupa más espacio en memoria?

```
struct informacion_celda a;
struct informacion_celda *b;
void main(){
int *p;
int a;
p = &a;
*p = 5;
}
```

iii. ¿Qué tamaño tienen las siguientes variables?

```
struct informacion_celda *ptr1, *ptr2;
struct informacion_operador *i1, *i2;
```

iv. Si una variable de tipo struct informacion_celda está almacenada en la posición de memoria 100, ¿qué dirección tienen cada uno de sus campos?

v. Si una variable de tipo struct informacion_celda * está almacenada en la posición de memoria 100, ¿qué dirección tiene cada uno de sus campos?

b. A partir de la declaración de las siguientes estructuras realice un programa en C que lea el array “ats” y devuelva los datos (nombre, país, deporte) del atleta que ha ganado mayor numero de medallas.

```
struct datos
{
	Char nombre[40];
	Char país[25];
};
struct atleta
{
	Char deporte[30];
	Struct datos pers;
	Int nmedallas;
};
struct atleta ats[30];
```

c. Desarrolle un programa en lenguaje C que incluya lo siguiente:
   i. Definición de una estructura adecuada para almacenar datos de un alumno (nombre, apellido, legajo, notas, condición). La condición de los alumnos será: LIBRE, REGULAR, PROMOCION PRACTICA, PROMOCION TEORICA, APROBACION DIRECTA. Este dato no será cargado por el usuario, sino que será determinado de manera automática por el programa. Utilice como guía el reglamento de la catedra Informática II
   ii. Permitir cargar una lista de alumnos con sus datos y las notas correspondientes a una determinada asignatura en un arreglo de la estructura definida en 1. Debe implementarse algún mecanismo (mediante menú) para permitir al usuario cargar los alumnos y detener la carga en cualquier momento. Se podrán cargar como máximo 50 alumnos.
   iii. Deberá ofrecer la posibilidad de modificar los datos de cualquier alumno que ya haya sido cargado
   iv. Finalizada la carga, el programa deberá ofrecer la opción de mostrar en pantalla el listado completo de alumnos y sus datos o mostrar un resumen indicando el porcentaje de los alumnos en cada condición.
   v. Utilizar comentarios para explicar el funcionamiento de cada funcion y otras partes relevantes del código implementado
   vi. Implementar los controles que considere necesario tales como verificar valores validos en la carga de datos, permitiendo re introducirlos tras un error

d. Defina la estructura `tiempo` que permita almacenar valores de tiempo expresados en horas, minutos y segundos. Luego, realice un programa que permita cargar un listado de valores de tiempo introducidos por teclado. El programa deberá solicitar al usuario el número de valores a recibir y reservar de manera dinámica la memoria para almacenarlos. Finalmente el programa deberá calcular y mostrar en pantalla el tiempo total dado por la suma de todos los valores ingresados.

```c
#include <stdio.h>
#include <stdlib.h>
struct tiempo{
	int horas; int minutos; int segundos;
};
int main()
{	int N = 0;
	printf("Ingrese cantidad de valores: ");
	scanf("%d",&N);
	//reservo memoria dinamica
	struct tiempo * tiempos = (struct tiempo *) malloc(N*sizeof(struct
tiempo));
	struct tiempo * tiempos = (struct tiempo *) calloc(N,sizeof(struct
tiempo));
	if(tiempos != NULL){
		//printf("Ingrese datos:");
		for(int = 0; i < N; i++)		{
			tiempos[i].horas
			(tiempos+i)->horas
			*(tiempos+i).horas
			printf("ingrese hora:");
			scanf("%d", &(tiempos[i].horas))
			printf("ingrese minutos:");
			scanf("%d", &(tiempos[i].minutos))
			printf("ingrese segundos:");
			scanf("%d", &(tiempos[i].segundos))
		}
		///sumar todos los tiempos
		struct tiempo suma = 0;
		for(int = 0; i < N; i++)		{
			suma.horas += tiempos[i].horas;
			suma.minutos += tiempos[i].minutos;
			suma.segundos += tiempos[i].segundos;
		}
		while(suma.segundos > 60)		{
			suma.segundos -=60;
			suma.minutos +=1;   }
		//mostrar resultado
		free(tiempos);
	}
	else
		return 1;
	return 0;
}
```