Apellido, Nombre:
Padrón:

Fundamentos de Programación

Exámen parcial - 08/05/2025

Tema Carl

Condición de aprobación: Se debe tener al menos 1 (un) ejercicio con Bien en cada parte (teórica y práctica) y adicionalmente otro ejercicio con Regular o más.

Parte teórica

Ejercicio 1

a. Explicar el algoritmo de inserción ordenada con tus palabras, de manera tal que una persona sin conocimientos de programación lo pueda entender. No explicar en forma de código ni usar ejemplos.
b. ¿Siempre es conveniente usar la inserción ordenada? ¿Hay algún caso donde no lo sea? Justificar.

Ejercicio 2

Dado el siguiente fragmento de código (que sabemos que compila correctamente):

```
#include<stdio.h>
#include<string.h>
#define MAX 10

int main(){
    char nombre[MAX];
    strcpy(nombre, "Homero");

    char* p = nombre + 2;

    printf("%c", p[2]);
}
```

a. Explicar paso por paso que es lo que ocurre en la ejecución del programa. ¿En algún momento se realiza un acceso inválido a la memoria?

- Si la respuesta es no, ¿qué letra se espera ver por pantalla al final?

- Si la respuesta es si, ¿dónde ocurre el error?

b. ¿Qué podría pasar si se desreferencia un puntero sin antes haberlo inicializado?

Parte práctica

Ejercicio 1

El Gato es un ladrón que está aterrorizando a todo Springfield. La policia de Springfield está haciendo un análisis de las casas robadas hasta el momento para determinar en qué barrio atacará El Gato próximamente.

Se sabe que su modus operandi es romper la cerradura de una casa y robarle algo importante a la familia. Siempre trata de llevarse un elemento por cada miembro del grupo familiar, y cuando no puede, se roba la mascota.

Se pide:

Dada una matriz de casas que fueron robadas, y sabiendo que cada columna es un barrio, realizar una función que determine qué barrio (representado por su índice) aún no tuvo un robo por El Gato.
Una casa se representa con el siguiente struct:

```
typedef struct casa_robada {
    char elementos_robados[MAX_NOMBRE][MAX_ELEMENTOS];
    // "mascota", "saxofón", "horno", etc.
    int cant_elementos_robados;
    bool cerradura_forzada;
    char color_fachada[MAX_COLOR];
    int cant_familiares;
} casa_robada_t;
```

Ejercicio 2

Lisa está haciendo un curso de criptografía, y cuando Homero le pidió una contraseña para su nueva computadora, utilizó el método de secretos que aprendió en el curso. Teniendo los secretos, ella puede siempre que necesite acceder a la contraseña, por si Homero se la olvida.

Un secreto se representa de la siguiente forma:

```
typedef struct secreto {
    char palabra_secreta[MAX_PALABRA];
    int codigo;
    int certificado;
} secreto_t;
```

Crear una función recursiva que, dado un vector de secretos, llene un string con la contraseña de Homero.

El método de encriptado consta de 2 partes:

- los secretos se recorren de una manera especial: siempre se comienza del primero y el índice del siguiente secreto a ver es el código dividido por el certificado del secreto actual.
- cada letra de la contraseña consistirá de la letra de la palabra secreta que esté en la posición del código.

Aclaración

Cuando el certificado es -1, significa que la letra en esa posición es la última letra de la contraseña.

Ejemplo:

posición	**0**	**1**	**2**	**3**
palabra	"placa"	"raton"	"isla"	"silla"
código	3	1	1	4
certificado	1	-1	1	2

- La primera letra de la contraseña se obtiene con la palabra y el código del elemento de la posición 0. En este caso es 'c'.
  El índice que sigue es código/certificado, es decir 3/1 = 3.
- La segunda letra se obtiene con el elemento en la posición 3. En este caso va a ser 'a'. El siguiente índice es 4/2 = 2.
- La tercera letra será entonces 's', porque la palabra en la posición 2 es "isla" y el código 1. La siguiente posición va a ser 1/1 = 1.
- La última letra de la contraseña será entonces 'a' porque es la que está en la posición 1 de la palabra "ratón".
  Como se llegó a un elemento con certificado igual a -1, ya se terminó de recorrer el vector y entonces la contraseña es "casa".