Fundamendez - 2C 2026 - Teórica semana 7

Ejercicios de repaso para el parcial

—

Ejercicio 1

Bart recorre los stands de la Convención de Springfield porque quiere recolectar firmas de todos los famosos que están allí.
La convención puede ser representada por una matriz de stand_t, este siendo:

```
typedef struct stand {
    bool esta_firmando;
    char personaje[MAX_NOMBRE];
    int rareza;
    int popularidad;
} stand_t;
```

Un autógrafo puede ser recolectado si:

- En el stand están firmando
  y
- La suma de las coordenadas del stand es par.

El valor de una firma se calcula como la rareza multiplicado por la popularidad dividido 10.

Se pide:
Implementar una función recursiva que reciba la matriz y devuelve el nombre del personaje con firma de mayor valor que se pueda recolectar.

Ejercicio 2

Dada la lista de números [100, 22, 28, 33, 95, 45, 46, 71, 83, 93] que se quiere ordenar de manera ascendente.

¿Cuál/es de los tres métodos de ordenamiento vistos en clase ordena correctamente la lista en menos iteraciones? (sin tener en cuenta las iteraciones extras propias de los algoritmos una vez ordenada la lista).

Justifique su respuesta.

(Aclaración importante: esta rta. está incompleta, debe acompañar de explicación que durante la clase se hizo oralmente).

Burbujeo

1) [22, 28, 33, 95, 45, 46, 71, 83, 93, 100]
2) [22, 28, 33, 45, 46, 71, 83, 93, 95, 100]

Selección

1) [93, 22, 28, 33, 95, 45, 46, 71, 83, 100]
2) [93, 22, 28, 33, 83, 45, 46, 71, 95, 100]

2 pasadas y no está ordenado → Burbujeo mejor por ahora

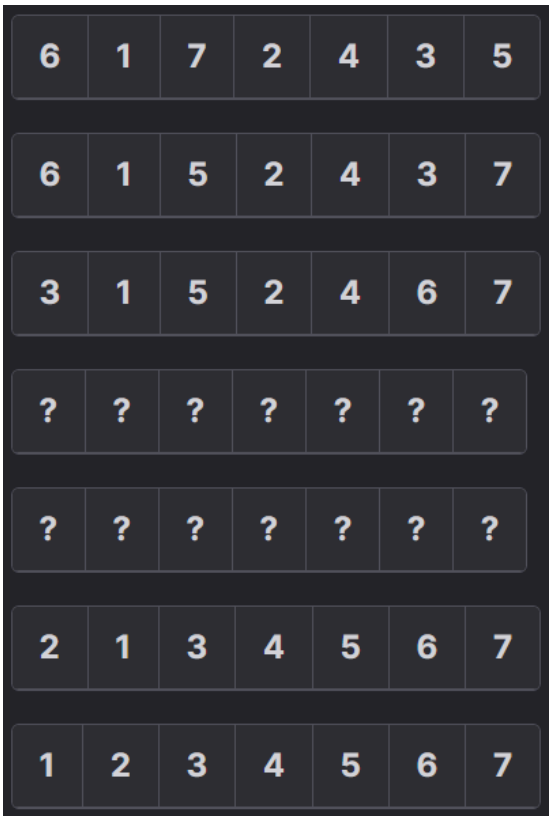
Inserción

1) [22, 100, 28, 33, 95, 45, 46, 71, 83, 93]
2) [22, 28,100, 33, 95, 45, 46, 71, 83, 93]
3) [22, 28, 33,100, 95, 45, 46, 71, 83, 93]

+2 pasadas y no está ordenado → Burbujeo wins

Ejercicio 3

a. Se tiene un programa que ordena un vector. Podemos ver el estado del vector en cada paso:



¿Cuáles son los dos estados faltantes del vector? ¿Qué método de ordenamiento se está usando? ¿Qué esperarías ver si el método fuera otro?

Ejercicio 4

a. Dada la lista de números [100, 92, 78, 33, 25, 25, 16, 11, 3, 1], ¿qué algoritmo usaría para buscar un elemento?
b. Y si la lista fuera [100, 22, 28, 33, 95, 45, 46, 71, 83, 93], ¿cuál usaría?
c. ¿Cambia algo de sus respuestas si van a tener que buscar muchísimas veces en el mismo vector del ítem b?

Ejercicio 5

a. Explique con sus propias palabras el algoritmo de búsqueda binaria, sin ejemplificar.
b. ¿Cuáles son las pre y post condiciones de la búsqueda binaria? ¿Qué beneficios tiene sobre la búsqueda lineal?

a) La búsqueda binaria es un algoritmo que busca un elemento en un conjunto ordenado.

Obtengo la posición de inicio y final del conjunto, y calculo el medio del conjunto para comparar el elemento buscado con el elemento en ese medio.

Si el elemento que está en el medio es el que yo busco, obtengo esa posición y termina el algoritmo.

Si no, si el elemento del centro es mayor al buscado, descarto los elementos mayores al elemento del centro (los siguientes si está ordenado de menor a mayor, o los anteriores si está ordenado de mayor a menor), y si es menor descarto los menores a esos (los siguientes si está ordenado de mayor a menor, o los anteriores si está ordenado de menor a mayor).

Vuelvo a calcular el medio del subconjunto que no descarté, con el nuevo inicio y fin de este subconjunto, y repito la comparación, hasta que se encuentre el elemento buscado, o que el inicio y fin sean iguales o absurdos.

Ejercicio 6

Dados los siguientes structs que sirven para representar a una persona y sus hobbies:

```c
typedef struct hobbie {
    char nombre[MAX_NOMBRE];
    char dias_de_la_semana[MAX_DIAS][MAX_NOMBRE];
    int cant_dias;
} hobbie_t;

typedef struct persona {
    char nombre[MAX_NOMBRE];
    hobbie_t hobbies[MAX_HOBBIES];
    int cantidad_hobbies;
} persona_t;
```

Determinar si los diferentes accesos a los campos de los mismos son válidos o no. En el caso de ser válidos decir a qué tipo de dato se está accediendo. En caso de ser inválido explicar por qué.

Aclaración: suponer que todos los campos están inicializados correctamente y que todos los arreglos tienen al menos un elemento.

1. persona.hobbies.nombre (inválido: no se indica a cual elemento del vector de hobbie_t s se quiere acceder).

2. persona.hobbies[0].dias_de_la_semana[persona.hobbies[0].cant_dias - 1][0] (válido: el tipo de dato al que se accede es char).

3. persona.hobbies[persona.cantidad_hobbies - 1].dias_de_la_semana (válido: el tipo de dato al que se accede es es char*/matriz de caracteres/vector de strings).

Ejercicio 7

a. Explicar qué son y para qué sirven las pre y post condiciones.
b. Además, realizar las pre y post condiciones de la siguiente función teniendo en cuenta que se encarga de repartir donas según el nombre de la persona.

   Pre:

   - 'nombre' debe ser un string (terminar en \0).
   - 'total_donas' debe ser distinto de 0.

   Post: Devuelve la cantidad a repartir según el nombre.

```c
float donas_por_nombre(char* nombre, int total_donas) {
    int letras = strlen(nombre);

    return (float)(letras * 2) / total_donas;
}
```

Ejercicio 8

En su espléndida trayectoria como niña nerd, Lisa ha ganado muchísimos trofeos en competencias educativas.

Un trofeo ganado por Lisa puede representarse con el siguiente registro:

```c
typedef struct trofeo {
    char nombre_competencia[MAX_NOMBRE];
    int fecha; // formato aaaammdd
    int puesto;
} trofeo_t;
```

Tiene una repisa dedicada exclusivamente a la exhibición de los mismos, y se encuentra ***ordenándolos por nombre alfabéticamente***, y en caso de tener 2 o más con el mismo nombre ***sub ordena por fecha del más reciente al más viejo***.

Se pide:
Crear un procedimiento que dada la repisa de Lisa (un vector de trofeo_t) lo ordene de la manera que ella necesita.