TEMA EVA | Apellido, Nombre: Padron:

Parte teórica

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

Ejercicio 1

WALL-E y EVA están jugando a las cartas. A EVA le gusta tener las cartas bien ordenadas, de menor a mayor por su número. Ella siempre ordena su mano de cartas usando el método de selección.

1. Explicar con tus palabras el método de ordenamiento de **selección**.
2. Ejemplificar ordenando el vector de números [10, 12, 5, 5, 2, 8, 1, 9] de manera ascendente.

Ejercicio 2

WALL-E no fue responsable y se enfermó con un virus (esto con Linux no le pasaba). El virus encriptó algunas de sus memorias con una función extraña y necesita descifrar la función para volver a recuperarlas. La función que armó el virus no cumple con las buenas prácticas entonces es difícil entenderla.

La función en cuestión es:

```
int virus_malicioso_pi_pi(int vec[], int tope){
      int a;
      for(int i = 0; i <= tope; i++){
      if(vec[i] < 100){
            a = vec[i];
      }
      }
            return a;
}
```

1. ¿Qué problemas podés encontrar en esta función?
2. Reescribir la función incorporando mejoras para los errores encontrados.
3. Explicar detalladamente qué hace la función arreglada y qué devuelve.

TEMA EVA | Apellido, Nombre: Padron:

Parte práctica

Ejercicio 1

En el mundo de Monsters Inc., las puertas desempeñan un papel fundamental en la recolección de sustos y gritos de niños asustados que alimentan la energía de todo el reino de los monstruos. En ese intrigante universo, cada puerta es única, tiene su propio color y personalidad. Todas las puertas tienen una luz que indican si está en condiciones para ser usada.

```
typedef struct puerta {
    bool luz_prendida;
    char color[MAX];
    int grito_acumulado;
} puerta_t;
```

Aclaraciones

- Si la luz no está prendida no se deberían tener en cuenta los otros campos.

Implementar una funcion que reciba una matriz de puertas y devuelva la **columna** con la mayor sumatoria de gritos acumulados por todas sus puertas de color “rojo”.

Ejercicio 2

Los enanitos están jugando al juego de la pelota que quema. Se ponen en ronda y se pasan la pelota entre ellos lo más rápido posible sin repetirse.

```
typedef struct enanito {
    char nombre[MAX_ENANITO];
    int proximo_enanito;
    int fuerza;
} enanito_t;
```

Dado un vector de **enanito_t** crear una función **recursiva** que simule la ronda de pases de pelotas y devuelva la **posición en el vector** del enanito que la pasó menos fuerte.

1. Se pide hacer un algoritmo que, dada la matriz mencionada, devuelva el enanito que sume más puntos contando todas las partidas.

Aclaraciones

- El último enanito de la ronda tendrá un -1 en el campo **proximo_enanito**.
- La ronda siempre arranca con el primer elemento del vector.
- Puede ser que alguien en el vector nunca reciba la pelota.

Ejemplo

Nombre: Gruñón	**Nombre: Feliz**	**Nombre: Dormilón**	**Nombre: Sabio**
Próximo: 3	Próximo: 0	Próximo: -1	Próximo: 2
Fuerza: 29	Fuerza: 10	Fuerza: 19	Fuerza: 33

La ronda será: Gruñón, Sabio, Dormilón. Feliz no recibe la pelota.
La función devolverá 2 que es la posición de Dormilón.