Apellido, Nombre:
Padrón:

Fundamentos de Programación

Exámen final - 12/12/2024

Condición de aprobación: Tener 2 ejercicios **Bien**.

Aclaraciones:

- Un ejercicio práctico se considera **Bien** cuando además de cumplir con lo pedido, aplica las buenas prácticas profesadas por la cátedra.
- En un ejercicio la parte práctica equivale al 75% de la calificación de ese punto mientras que la parte teórica equivale al 25%.

Ejercicio 1

Parte práctica

Annie y Hallie siguen con la ardua tarea de hacerle bromas y maldades a Meredith para que el plan de reunir a sus padres resulte exitoso.
En un principio crearon un archivo con todas las bromas que se les ocurrieron, donde cada línea es la descripción de una broma.
Cada día realizan varias de estas bromas y las quitan del archivo, quedando las que aún no pusieron en práctica.

Se pide

Dado dos archivos, ***bromas_no_realizadas.txt*** y ***bromas_del_dia.txt*** actualizar el primer archivo quitando las bromas que están en el segundo.

Parte teórica

Nombrar y explicar brevemente los tres métodos de ordenamientos dictados en la cátedra. ¿A qué debe cada uno su nombre?

Ejercicio 2

Parte práctica

Jane se encuentra perdida en la selva y ¡rodeada de animales!
De Tarzán aprendió que un animal es peligroso y no debe acercarse a él si:

- Es un animal no hervívoro que tiene 3 o más dientes filosos.

  o

- Es un animal no hervívoro que tiene escamas y es venenoso.

Un animal puede ser representado con el siguiente struct:

```
typedef struct animal {
    char alimentacion; // 'C' (carnivoro), 'H' (hervivoro) u 'O' (omnivoro)
    char piel[MAX_PIEL]; // "pelaje", "escamas" o "plumaje"
    dientes_t dientes;
    bool venenoso;
} animal_t;

typedef struct dientes {
    int cantidad_dientes;
    bool dientes_filosos[MAX_DIENTES];
} dientes_t;
```

Se pide

Crear una función que reciba una matriz de animales que representa la selva y una coordenada que representa la posición en la que Jane se encuentra, y devuelva la cantidad animales peligrosos que tiene alrededor (arriba, abajo, izquierda, derecha y sus cuatro diagonales):

Aclaraciones

- No se debe tener en cuenta el animal que se encuentra en la misma posición que Jane.

Parte teórica

1. ¿Cuál es la diferencia entre el tope y el máximo de un vector? ¿Qué representa cada uno?

Ejercicio 3

Parte práctica

En Monster Inc quieren saber si Mike efectivamente hizo su papeleo. Tenía que firmar y ordenar las en una pila o torre, para que sus supervisores lo pudieran firmar y guardar. Como Mike ordena sus reportes en una pila, solo puede agregar o sacar de arriba.

Se tiene un TDA "papeleo" que guarda los IDs de los reportes (enteros) en su escritorio. El TDA cuenta con las siguientes primitivas:

```
// Pre: -
// Post: Crea un TDA papeleo en el heap y devuelve un puntero al
// mismo. Devuelve NULL si no lo pudo crear.
papeleo_t *papeleo_crear();

// Pre: El papeleo recibido fue previamente creado con `papeleo_crear`.
// Post: Libera la memoria que se reservó en el heap para el papeleo.
void papeleo_destruir(papeleo_t *papeleo);

// Pre: El papeleo recibido fue previamente creado con `papeleo_crear`.
// Post: Se eliminará el primer reporte del papeleo. Devuelve true si se
// eliminó correctamente o false si el papeleo estaba vacío y no se pudo
// eliminar nada. En caso de devolver true, el elemento que se acaba de
// eliminar será devuelto por referencia mediante el parámetro
// `elemento_eliminado`.
bool papeleo_eliminar_primero(papeleo_t *papeleo, int *elemento_eliminado);

// Pre: El papeleo recibido fue previamente creado con `papeleo_crear`.
// Post: Agrega un reporte al principio del papeleo.
bool papeleo_agregar_primero(papeleo_t *vector, int nuevo_elemento);
```

Hacer una función que reciba un `papeleo_t` creado, que elimine todos los elementos del papeleo y que devuelva `true` si los mismos estaban en orden descendente. Cuando se hayan quitado todos los elementos tiene que destruirse el `papeleo_t` .

Aclaraciones

- No es necesario que todos los IDs del papeleo sean contiguos pero si que estén en orden descendente. Por ejemplo 10 -> 8 -> 2 estaría permitido.

Parte teórica

1. ¿Qué es un TDA y para que sirven?
2. Crear la firma y las pre y post condiciones de una función del TDA "papeleo" que sirva para contar la cantidad de elementos que contiene. No hace falta implementarla.