Fundamendez - 1C 2025 - Clase 14 (martes)

TDAs - Pila y Cola

Ejercicio pila (Final 1 2C 2024)

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:

```c
// 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***.

Ejercicio cola

Una entrega de TP en fundamendez puede ser representado por el siguiente registro:

```
typedef struct entrega {
   bool pasan_todas_pruebas;
   bool pasan_pruebas_minimas;
   char
buenas_practicas_utilizadas[MAX_BUENAS_PRACTICAS][MAX_NOMBRE_BUENA_PRAC
TICA];
   int cantidad_buenas_practicas_utilizadas;
} entrega_t;
```

Se tiene un TDA cola que tiene las entregas recibidas por algotrón (patente pendiente). Se pide:

- Crear una función que reciba un puntero a la cola cargada con las entregas y devuelva el porcentaje de TPs que resultaron aprobados.
- Crear una función que quite de la cola las entregas desaprobadas manteniendo el orden.

Aclaración: Un TP está aprobado si pasan todas las pruebas, tiene 4 o más buenas prácticas utilizadas y una de ellas es “Modularizacion”.

El TDA cola tiene las siguientes primitivas:

```c
/* Pre condiciones:
* Post condiciones: Crea una cola vacía. Devuelve NULL en caso de error.
*/
algotron_t* algotron_crear();


/* Pre condiciones: algotron debe apuntar a una cola creada con
`algotron_crear`.
* Post condiciones: Encola una entrega. Devuelve true si pudo encolar,
false en caso contrario.
*/
bool algotron_encolar(algotron_t* algotron, entrega_t nueva_entrega);


/* Pre condiciones: algotron debe apuntar a una cola creada con
`algotron_crear`.
* Post condiciones: Desencola el primer elemento de algotron.
                    Devuelve true si se pudo desencolar, false si la cola
estaba vacía.
*/
bool algotron_descolar(algotron_t* algotron);


/* Pre condiciones: algotron debe apuntar a una cola creada con
`algotron_crear`.
* Post condiciones: Devuelve el primer elemento de algotron sin
desencolarlo. O -1 en caso de error o cola vacía.
*/
entrega_t algotron_primero(const algotron_t* algotron);


/* Pre condiciones: algotron debe apuntar a una cola creada con
`algotron_crear`.
* Post condiciones: Devuelve true si algotron está vacío, false en caso
contrario.
*/
bool algotron_esta_vacio(const algotron_t* algotron);


/* Pre condiciones: algotron debe apuntar a una cola creada con
`algotron_crear`.
* Post condiciones: Destruye algotron y libera la memoria.
*/
void algotron_destruir(algotron_t* algotron);
```