Algoritmos y Estructuras de Datos II

Examen final de Laboratorio – LIBRES.

Se deberá implementar el *TAD BallotBox* (urna) especificado en el exámen teórico.

La estructura interna deberá contener un arreglo de contadores donde se guarda la cantidad de votos de cada candidato. Los candidatos se representan con posiciones válidas de ese arreglo, por ejemplo:

votes → [4, 0, 8]

El candidato 0 tiene *4* votos, el candidato 1 tiene *0* votos, y el candidatos 2 tiene *8* votos.

Además se deberá mantener dentro de la estructura la cantidad de votos total y la cantidad de votos en blanco de la urna.

Ejercicio 1:

Dentro de *ballotbox.c, i*mplementar las funciones declaradas en *ballotbox.h:*

Crea una urna vacía preparada para *N* candidatos.
ballotbox_t ballotbox_empty(unsigned int *N*)

Agrega un voto para un candidato dentro de la urna.
ballotbox_t ballotbox_candidate_vote(ballotbox_t box, candidate_t c)

Agrega un voto en blanco dentro de la urna.
ballotbox_t ballotbox_blank_vote(ballotbox_t box)

Cancela un voto de un candidato. Si el candidato no tiene votos, esta función no hace nada.
ballotbox_t ballotbox_vote_cancel(ballotbox_t box, candidate_t c)

Verifica si la urna esta vacía.
bool ballotbox_is_empty(ballotbox_t box)

Devuelve la cantidad total de votos.
unsigned int ballotbox_total_votes(ballotbox_t box)

Devuelve la cantidad de votos en blanco.
unsigned int ballotbox_blank_votes(ballotbox_t box)

Verifica si el candidato dado tiene votos.
bool ballotbox_has_votes(ballotbox_t box, candidate_t c)

Verifica si el candidato dado es el ganador de la elección. Un candidato ganador (si existe) es aquel que tiene estrictamente más votos que todos los demás.
bool ballotbox_is_winner(ballotbox_t box, candidate_t c)

Verifica si la urna tiene un candidato ganador.
bool ballotbox_has_winner(ballotbox_t box)

Crea una urna nueva contando los votos de ambas urnas dadas como argumento.
No modifica las urnas originales.
No necesariamente las urnas que recibe tienen la misma cantidad de candidatos.
ballotbox_t ballotbox_join(ballotbox_t box1, ballotbox_t box2)

Imprime en pantalla una urna, mostrando la cantidad de votos de cada cantidato,
la cantidad de votos total y la cantidad de votos en blanco.
void ballotbox_print(ballotbox_t box)

Destruye una urna liberando sus recursos de memoria.
ballotbox_t ballotbox_destroy(ballotbox_t box)

Devuelve la máxima cantidad de votos obtenida por algún candidato.
unsigned int ballotbox_max_vote_count(ballotbox_t box)

Devuelve cuántos candidatos tienen una cantidad de votos no menor a la de ningún otro candidato.
Ejemplo: votes → [4, 4, 3, 2, 4] los candidatos 0, 1 y 4 son los mejores candidatos. La función devuelve 3.
unsigned int ballotbox_best_candidates_count(ballotbox_t box)

Devuelve un arreglo con aquellos candidatos cuya cantidad de votos no es menor a la de ningún otro candidato. En el ejemplo de la función anterior devuelve el arreglo [0, 1, 4]. El tamaño del arreglo está dado por la función anterior.
candidate_t *ballotbox_best_candidates(ballotbox_t box)

Ejercicio 2:

En *main.c*, implementar una función *main* con al menos un caso de test que utilice
todas las funciones anteriores.

Importante:

Se restarán puntos en caso de **memory leaks** o **invalid reads**.

No olvidar entregar el final de acuerdo a las instrucciones del profesor a cargo.