Lab #9

Spring 2025

Requirements

In this lab, you'll implement a simple system for tracking packages sorted by delivery distance. The goal is to perform fast searches and manage memory efficiently. You'll practice using structs, binary search, and dynamic memory allocation with embedded metadata.

```
typedef struct {
    unsigned int trackingNumber;
    float deliveryDistance;
    unsigned short weight;
} Package;
```

1.1 createArray

```
void * createArray(int arraySize, int elementSize);
```

Allocate memory for a dynamic array that holds metadata. Before the array, store the array size (number of elements) as an int. A pointer to the first actual element of the array (not the metadata). Returns **`NULL`** if allocation fails.

1.2 arraySize

```
int arraySize(void * array);
```

Returns the number of elements stored in the array, which is stored just before the start of the array pointer. The input must be a pointer to an array allocated using **`createArray`**.

1.3 findPackage

```
int findPackage(Package * array, Package * query);
```

Search the sorted array recursively using binary search (based on **`deliveryDistance`**). The index of the matching **`Package`** if found; -1 otherwise. The complexity must be O(log n)

1.4 comparePackages

```
int comparePackages(Package *a, Package *b);
```

Compares two packages by their **`deliveryDistance`**. Returns negative if a < b, returns zero if a == b, and returns positive if a > b. The complexity must be O(1).

1.5 destroyArray

```
void destroyArray(void * array);
```

Properly frees the array allocated using **`createArray`**

Rubric

1. (2 pts) void * createArray(int arraySize, int elementSize);
   * (1 pts) Creates array successfully
   * (1 pts) Handles exceptions
2. (1 pts) int arraySize(void *array);
   * (1 pts) Returns the correct array size
3. (8 pts) int comparePackages(Package *a, Package *b);
   * (1 pts) returns negative if a<b
   * (1 pts) returns zero if a == b
   * (1 pts) returns positive if a>b
   * (5 pts) Follows O(1) complexity
4. (7 pts) int findPackage(Package *array, Package *query);
   * (1 pts) Return correct index number if found
   * (1 pts) Return -1 if not found
   * (5 pts) Follows O(logn) complexity
5. (2 pts) void destroyArray(void *array);
   * (1 pts) frees array correctly
   * (1 pts) Handles exceptions correctly