Worksheet 0: Building a Simple ADT Using an Array

In Preparation: Read about basic ADTs. In this worksheet we will construct a simple BAG and STACK abstraction on top of an array. Assume we have the following interface file "arrayBagStack.h"

```
----------
# ifndef ArrayBagStack
# define ArrayBagStack
# define MAX_SIZE 100


# define TYPE int
# define EQ(a, b) (a == b)

struct arrayBagStack {
TYPE data [MAX_SIZE];
int count;
};

/* Interface for Bag */
void initBag (struct arrayBagStack * b);
void addBag (struct arrayBagStack * b, TYPE v);
int containsBag (struct arrayBagStack * b, TYPE v);
void removeBag (struct arrayBagStack * b, TYPE v);
int sizeBag (struct arrayBagStack * b);

/* Interface for Stack */
void pushStack (struct arrayBagStack * b, TYPE v);
TYPE topStack (struct arrayBagStack * b);
void popStack (struct arrayBagStack * b);
int isEmptyStack (struct arrayBagStack * b);

# endif
----------
```

Your job, for this worksheet, is to provide implementations for the following operations.

```
/* Bag Implementation */
void initBag (struct arrayBagStack * b){

	assert(b != 0);
	b->count = 0;
}

void addBag (struct arrayBagStack * b, TYPE v) {

	assert(b != 0);
	if (b->count < MAX_SIZE)
	{
		b->data[count] = v;
		b->count++;
	}
}

int containsBag (struct arrayBagStack * b, TYPE v) {

	assert(b != 0);

	/* If array is not empty, search for the value in the array */
	if (b->count != 0)
	{
		for (int i = 0; i < b->count; i++)
		{
			/* If the value is found, return true */
			if (EQ(b->data[i], v))
			{
				return 1;
			}
		}
		/* If the value is not found, return false */
		return 0;
	}
	/* If the array is empty, return false */
	else
	{
		return 0;
	}
}

void removeBag (struct arrayBagStack * b, TYPE v) {
```

```
        assert(b != 0);

        /* If array is not empty, search for the value in the array */
        if (b->count != 0)
        {
                for (int i = 0; i < b->count; i++)
                {
                        if (EQ(b->data[i], v))
                        {
                                /* If it is found, replace with the last element value */
                                b->data[i] = b->data[b->count – 1];

                                /* Zero the last element, update count, and return */
                                b->data[b->count – 1] = 0;
                                b->count--;
                                return;
                        }
                }
        }
}

int sizeBag (struct arrayBagStack * b) {

        assert(b != 0);
        return b->count;
}


/* Stack Implementation */

void pushStack (struct arrayBagStack * b, TYPE v) {

        assert(b != 0);
        if (b->count < MAX_SIZE)
        {
                b->data[count] = v;
                b->count = b->count + 1;
        }
        /* Or just addBag(b, v) because the code is the same! */
}
TYPE topStack (struct arrayBagStack * b) {

        assert(b != 0);
        return b->data[b->count - 1];
```

```
}

void popStack (struct arrayBagStack * b) {

        assert(b != 0);
         if (b->count > 0)
        {
                b->count--;
        }

}

int isEmptyStack (struct arrayBagStack * b) {

        if (EQ(b->count, 0))
        {
                return 1;
        }
        else
        {
                return 0;
        }
 }
```