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

struct arrayBagStack {
      TYPE data [100];
      int count;
};

void initArray(struct arrayBagStack * b);
void addArray (struct arrayBagStack * b, TYPE v);
int containsArray (struct arrayBagStack * b, TYPE v);
void removeArray (struct arrayBagStack * b, TYPE v);
int sizeArray (struct arrayBagStack * b);

void pushArray (struct arrayBagStack * b, TYPE v);
TYPE topArray (struct arrayBagStack * b);
void popArray (struct arrayBagStack * b);
int isEmptyArray (struct arrayBagStack * b);
# endif
```

Your job, for this worksheet, is to provide implementations for all these operations.

```
void initArray (struct arrayBagStack * b){

      *b = malloc(sizeof(struct arrayBagStack)); //allocate
memory for struct

      assert(b ! = 0); // if b is null, terminate program
[#include asser.h]

      b->count = 0;



}
```

```
/* Bag Interface Functions */

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

      b->data[b->count] = v;
      b->count++;


}

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

      for (int i = 0; i < 100; i++) {
            if (b->data[i] == v)
                  return i;
            }

      return 0;

}

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

      TYPE temp;

      for (int i = 0; i < 100; i++) {
            if (b->data[i] == v) {
                  for (int j = i; j < 100; j++) {
                        temp = b->data[++j];
                        b->data[j] = temp;
                  }
                  b->data[b->count] = 0;
                  b->count--;
            }







}

int sizeArray (struct arrayBagStack * b) {

      return b->count;



}
```

```
/* Stack Interface  Functions */

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

      b->data[b->count] = v;
      b->count++;

}

TYPE topArray (struct arrayBagStack * b) {

      return b->data[b->count];
}

void popArray (struct arrayBagStack * b) {

      b->data[count] = 0;
      b->count--;


}


int isEmptyArray (struct arrayBagStack * b) {

      if (b->count = 0)
            return 1;
      else
            return 0;



}
```

A Better Solution…

This solution has one problem. The arrayBagStack structure is in the .h file and therefore exposed to the users of the data structure. How can we get around this problem? Think about it…we'll return to this question soon.

Encapsulate these functions in a struct and then create a ne2w data struct that implements these functions with its own interface functions