**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){

b->count = 0;

}

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

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

b->count++;

}

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

int i;

for(i = 0; i < b->count; i++)

{

if(b->data[i] == v) //if v is found, return 1 (true)

{

return 1;

}

}

return 0; //otherwise, return 0 (false)

}

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

int i, j;

if(b->count != 0) //if bag is empty, do nothing

{

for(i = 0; i < b->count; i++)

{

//if v is found, shift remaining array to the

//left one space and decrement count

if(b->data[i] == v)

{

//start with index to the right of i, copy

//its value to the left

for(j = i + 1; j < b->count; j++)

{

b->data[j - 1] = b->data[j];

}

b->count--;

}

}

}

}

int sizeBag (struct arrayBagStack \* b) {

return b->count;

}

/\* Stack Implementation \*/

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

assert(b->count < MAX\_SIZE);

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

b->count++;

}

TYPE topStack (struct arrayBagStack \* b) {

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

}

void popStack (struct arrayBagStack \* b) {

b->count--;

}

int isEmptyStack (struct arrayBagStack \* b) {

return b->count == 0;

}