Worksheet 0 - Group 1

Worksheet Group 1 Members

Marc Clinedinst: clinedim@onid.oregonstate.edu
Kelby Faessler: faesslek@onid.oregonstate.edu
James Fitzwater: fitzwatj@onid.oregonstate.edu
Tom Gariepy: gariepyt@onid.oregonstate.edu
Sean Reilly: reillys@onid.oregonstate.edu
Joseph Struth: struthj@onid.oregonstate.edu

Worksheet 0: Building a Simple ADT Using an Array

In this worksheet, our aim is to implement functions that control the behavior of a Bag and a Stack using an array of size 100. The functions are listed below, with comments where necessary.

```
/*
     The size of the array is defined in the structure
     definition--that is, it is set to 100. Thus, we only need
     to set the initial count for the array in this function, which
     is of course 0.
*/
void initArray(struct arrayBagStack *b) {
     b->count = 0;
}

/*
     We first check that the size of the array is less than 100. If
     the size of the array is less than 100, we add the value passed
     to the function at the appropriate index and increment the
count
     of values.
*/
void addArray(struct arrayBagStack *b, TYPE v) {
     if (b->count < 100) {
          b->data[b->count] = v;
          b->count++;
     }
}

/*
     Since the values in the array are not necessarily ordered, we
```

```
     perform a linear search to find the target value. If the target
     value is found, we return 1 to indicate a positive hit;
     otherwise, we return 0 to indicate that the value was not
found.
*/
int containsArray(struct arrayBagStack *b, TYPE v) {
     int index;

     for (index = 0; index < sizeArray(b); index++) {
          if (b->data[index] == v) {
               return 1;
          }
     }

     return 0;
}

/*
     In this function, we remove a target value from the array. We
     first determine whether the value is in the array.  If the
value
     is in the array, then we search for the index at which it
     occurs.  Once we have found this index, we shift every value
     to the left by one and decrement the count of values in the
     array.
*/
void removeArray (struct arrayBagStack *b, TYPE v) {
     int index;

     if (containsArray(b, v)) {
          for (index = 0; index < sizeArray(b); index++) {
               if (b->data[index] == v) {
                    break;
               }
          }

          if (index < (b->count - 1)) {
               for (; index < sizeArray(b) - 1; index++) {
                    b->data[index] = b->data[index + 1];
               }
          }

          b->count--;
```

```
	}
}

/*
	This function returns the size of the array.  It does so by
	returning the value stored in the count field.
*/
int sizeArray(struct arrayBagStack *b) {
	return b->count;
}

/*
	This function pushes a value to the top of the stack. Given
	that this behavior is the exact same as the addArray function,
	we needn't redefine this behavior.
*/
void pushArray (struct arrayBagStack *b, TYPE v) {
	addArray(b, v);
}

/*
	This function returns the value at the top of the stack.
*/
TYPE topArray (struct arrayBagStack *b) {
	return b->data[b->count - 1];
}

/*
	This function removes the value at the top of the stack. It
does
	so simply by decrementing the value of count, so long as count
	is greater than 0.
*/
void popArray (struct arrayBagStack *b) {
	if (b->count > 0) {
		b->count--;
	}
}

/*
	This function returns an integer value representing whether the
	array is empty. 0 indicates that the stack is empty; any other
	value indicates that the stack is non-empty.
```

```
*/
int isEmptyArray (struct arrayBagStack *b) {
      return b->count == 0;
}
```

In addition to the function definitions above, this worksheet also asked us to consider the downside of including the definition of the ArrayBagStack structure in the .h file. As the assignment description notes, this exposes the internal details of the ArrayBagStack to the user, which violates the principle of encapsulation. A better solution would be to add a type definition of ArrayBagStack to the .h file and move the actual definition of the ArrayBagStack to the .c file. This will resolve the violation of encapsulation.

Group Minutes

Our Minutes are available on Piazza at the following URL:

https://piazza.com/class/ib2kus4hsie528?cid=72