Hayley Yukihiro

CSIS215 – D01

November 9, 2020

**Programming Assignment 1: Bag Dictionary**

**Approach**

To start off the approach document, I will outline the direction of the approach to the bag dictionary programming assignment. Each step will be explained by function approach and solution:

Outline Approach

1. Abag.h Feature Implementation
   1. Constructors
   2. Destructor
   3. addItem
   4. find
   5. remove
   6. removeTop
   7. inspectTop
   8. emptyBag
   9. operator+=
   10. size
   11. bagCapacity
2. Abag.h Testing Sequence
   1. addItem
   2. bagCapacity
   3. size
   4. inspectTop
   5. find
   6. removeTop
   7. operator+=
   8. remove
   9. emptyBag
3. BDictionary.h Feature Implementation
   1. Constructors
   2. Deconstructor
   3. Clear
   4. Insert
   5. remove
   6. removeAny
   7. find
   8. size
4. BDictionary.h Testing Sequence
   1. insert
   2. size
   3. find
   4. remove
   5. removeAny
   6. clear

The preliminary steps were completed to begin the assignment by compiling all the required .cpp and .h files into a folder that is named per instructions.

**ABag.h Feature Implementation**

To begin the ABag.h implementation, I looked at bagADT.h to brainstorm ideas of how to implement the various functions in Abag.h. Before implementing the functions, I decide to start off Abag.h by establishing the constructor and destructor.

**Constructor: Initialize objects in the class Abag**

* Since the bag works like a stack, I decided to utilize the example of Array-Based stacks since the bag is like a stack and is array-based. I debated on whether to use global variables for the constructor, but I decided to set a parameter for the constructor that states the maximum size of the array which I wrote out as 20 since the dictionary size in the bagtestmain.cpp is 20.
* Capacity is set to the maximum size of the array which is 20 from the parameter
* The used data is initialized to zero since the array has no items in it at the start
* The data array is declared of fixed size when the bag is created

**Destructor: Destructs objects in the class Abag**

* The items in the array data are deleted and based off AStack example

**addItem: Add items into the bag**

* The bag is like a stack since it fills from bottom to top. Therefore, I implemented the code like AStack in the textbook that adds or “pushes” an item into the bag.
* I implemented addItem with a bool format that checks if the bag is full or not. If it is not full, then the item is added to the bag. No errors were risen at this point.

*Test:*

* + All the test functions in the bagtest.cpp were for BDictionary functions so I wrote a test for the addItem function.
  + I created a test environment by declaring myBag to have a capacity of 12 which is more than 10 but not to the max of 20.
  + I created a test KV pair with an integer and string.
  + I created a test output which shows that one item was added to the addBag from 0 to 1.

**remove: Removes items from the bag**

* Remove is where the bag implementations differ from the stack model because the bag allows the removal of items that are not at the top of the stack. Therefore, the pop method should not be used for this implementation.
* The remove function will be a bool like the addItem because it will need to check if there are any items to be removed. If there is an item to be removed, it will come back as true and proceed.
* I based my code off the array-based list implementation example from the textbook. Once it updates the search item to the bag item, it proceeds to remove the item by going back through the bag and shifting all the items. By shifting the items, the item is removed.
* Once the shifting is complete, the bag is updated by decrementing the size of the bag so it fully accounts for the removal.
* I left the return false statement outside the for loop for finding the item match in the bag because I do not want it to keep returning false in the loop. Once the for loop runs its course in the search, it will just return false.
* After testing, the remove was not removing the item. I went back and debugged to find that I need to assign the true and false returns to a variable so it doesn’t cut out of the loop early before the decrement.

**removeTop: Removes the top item from the bag**

* This implementation is similar to popping from the top of the stack which is demonstrated in the AStack code of the textbook
* Unlike AStack, the implementation is bool so I formatted the function to check if the current number of items equal zero. If they do not count to zero, then there is an item to be removed. If there is zero items in the bag, the code will return false and remain unchanged.

**find: Finds an item in the bag specified by the parameter**

* As noted in bagADT.h, the find function is similar to remove but there is no remove operation
* I copied the remove code, replaced item with returnValue, and deleted the removal portion.

**InspectTop: Inspects the top of the bag**

* Similar to removeTop function but there is no removal of the top
* Since the InspectTop is similar to a stack function, I based the code off of topValue from the AStack example of the textbook

**emptyBag(): Empties the bag items**

* Initially based my code off of the AList example from the textbook that shows code on how to clear a list array
* Since the bag is a like a stack, I opted to the AStack example of setting the current item count of the bag to zero to empty all the contents.

**operator+=: use the += operator to add an item to the bag**

* Overloaded the operator by having it call the addItem function for addend when the operator is used to add an item to the bag

**size: get the size of the bag**

* Returns used which holds the count of current items in the bag

**bagCapacity: get the capacity of the bag**

* Returns capacity which contains the number of items the bag can hold

**Abag.h Test Sequence**

After the functions are implemented in the header, each one will be tested in bagtestmain.cpp to ensure functionality before implementing BDictionary.h which utilizes Abag.h functions. All the test functions in the bagtest.cpp were for BDictionary functions so I wrote a test for the addItem function. I created a test environment by declaring myBag to have a capacity of 12 which is more than 10 but not to the max of 20. I created a test KV pairs with an integer and string. I made about 13 which is one more than the declared capacity for the bag.

* **addItem**
  + I created a test output which shows that one item was added to the addBag. The output should be 1 because it is adding 1 item for each KV pair.
  + By observing the locals window, I can see that the used variable is progressed to 12 which shows that all 12 items were successfully added to the bag to full capacity.
  + When I added the 13th item, the addItem did not add it to the bag by outputting zero because the bag reached capacity at 12.
* **bagCapacity** 
  + I created a test output that calls the bagCapacity function and outputs the current maximum bag capacity which should be 12.
* **size**
  + I created a test output that calls the size function and outputs the amount of items currently in the bag which should be 12.
* **inspectTop**
  + I created a test output that calls the insepctTop function and outputs the top item of the bag
  + The bag is a stack where the first item is at the bottom and the last item is at the top. So the output should be 1 item and the top item should be Coffee which was the last KV pair added to the bag.
* **find**
  + I created a test output that calls the find function and outputs the item found
  + For the key and value pairs, the key is the search code which brings up the value.
  + I created three key and value pairs that need to be found. The first one and second ones should be in the bag while the third pair is not in the bag.
  + I tested for the find function to find the key and the value for each item.
  + I outputted the first item found which should be 1 item. Then I outputted the value of the found item which should be cereal since we searched for the value with the key 3.
  + I outputted the second item found which should be 1 item. Then I outputted the key of the found item which should be 6 for Sunkist gummies.
  + I outputted the second item but none which should be 0.
* **removeTop**
  + I created a test output that calls the removeTop function
  + After calling the function with the myBag parameter, I outputted the return of the key and value for the item at the top being removed which should be 12 and Coffee.
  + I outputted the size function to make sure the item was removed. The size went from 12 to 11.
* **operator+=**
  + I created a test output that calls the operator += to add a new item to the bag
  + I created a key and value pair that will be added using the operator
  + I check if the pair was added correctly by calling inspectTop because it was the most recent addition. The output should be one on top.
  + Check that the top value is the key and value of the new pair which should be 15 and Talking Fishbone
  + I checked if the size of the bag was updated by calling the size function. The bag should be back to 12.
* **remove**
  + I created a test output that calls the remove function to remove one requested item from the bag
  + Checked if the correct item was removed by checking if the code identified it is removing one item. Then I outputted the key and value of the removed item to make sure it matches the requested remove item pair.
  + Check the size of the bag which should be 11 instead of 12.
  + I adjusted the remove function because it broke out of the loop early when I just wrote return true. So there was no decrementing and the size remained 12. I adjusted this error by assigning the true and false return values to a variable so the function could continue its loop.
* **emptyBag**
  + I created a test output that calls the emptyBag function to clear all contents in the bag
  + Checked the current item count in the bag by calling the size function. The count should be zero for used.

**BDictionary.h Implementation**

To begin the BDictionary implementation, I studied the dictionaryADT to get an understanding of what the functions should do. I brainstormed and looked into the textbook to find ways to complete the functions using the ABag functions. I started the process by creating a constructor and destructor for the class.

**Constructor: Initializes objects in the BDictionary class**

* I referenced the unsorted array-based dictionary example from the book to help build the constructor for this class
* I instantiated the dictionary object by allocating storage space for the size of the dictionary which is not specified and labeled as defaultSize.

**Destructor: Destructs objects in the BDictionary class**

* Deleted the dictionary object which I learned from the UALdict example in 4.4 of the textbook

**clear: Reinitialize dictionary**

* By calling dictionary’s Abag function emptyBag(), I was able to implement it into the clear function to reinitialize the dictionary object. I was able to reference the code from the textbook example UALdict.

**insert: Insert a record**

* Similar to the unsorted array-based dictionary example from the textbook, I was able to loosely base the Boolean insert function from it. From this reference, I was able to understand how to utilize the addBag function to perform the insertion.
* Since the function needs to return a true or false response, I created a bool variable that will be assigned to the addBag function call
* If the addBag function is unsuccessful at insertion the new pair, the variable will be false and deliver an error message.

**remove: Removes record from the dictionary**

* Creates a temporary pair that contains the rtnVal that will be used as a parameter to search for the item and remove it by calling the remove function from ABag.h.
* Needed to create a temporary key for the missing key value in order for the KV pair to instantiate
* I assigned the outcome to a bool variable. Once the function returns its value, if true, it will assign the rtnVal to the value from the temp pair after the function call and return that its true. If the function returns false, it will output an error message.

**removeAny: Take an arbitrary record from the dictionary and removes it**

* Created a temp variable and place it in the function call removeTop from ABag.h
* Assigned the result to a bool variable that if true, will update the returnValue and return true. If false, it will output an error message.

**find: Looks for a record using the key**

* Create temp variable and place it in the function call find from ABag.h
* Assigned result to bool variable that if true, will update the returnValue and return true. If false, will produce an error message.

**size: Return the number of records in the dictionary**

* Calls the size function from ABag.h for dictionary object

**BDictionary Testing Sequence**

The test sequence for all the BDictionary.h functions are existing in the bagtestmain.cpp. The first test was in myIntStrDict tests and the second test was myStrIntDict tests:

* Unable to being the program due to the key pair unable to instantiate
* I realized that in my remove function, I did not assign any temp k value so I had to create a private temp key. From there, I was able to reassign the temp pair which allowed the program to run smoothly.
* All the tests ran as expected.