Pre-Programming

3/22/2023:

1. There are several c++ concepts listed in the assignment instructions I am unfamiliar with.
    a. Smart Pointer (review videos)
    b. KVPairs (looks like it's just a stand-alone header file based on dictionary concepts)
    c. Array-based implementation (I know what an array is in terms of lists, but not this exact wording)

3/23/2023:

1. Refreshed on C++ pointers
2. Learned about Smart Pointers
3. Instructor replied to help email saying the following about the bag project:
    a. The assignment is implementing a data structure that is called a Bag Dictionary, and the material describes what it is (probably module 2)
    b. The ABag.h and BDictionary.h files need to be modified.
4. Read the textbook material and took notes for some of module 2
    a. Array-Based Implementation was covered

3/24/2023:

1. Finished the textbook material
    a. Crucial knowledge on dictionary type
    b. The "stack" from chapter 4 looks and sounds a lot like the bag data type (only accepting input from the end or "opening") . I can use this to deploy the bag methods.
2. Began writing comments on the bagtestmain.cpp file to help me understand what each line is doing.
3. Refreshed on some C++ concepts to help me with finer details (Const & static keywords mainly)
4. Rewatched video on smart pointers, took notes on smart pointers

Programming Approach:

3/27/23:

1. Wrote all Bag methods in the ABag.h file by commenting out everything having to do with the dictionary methods, as well as every bag method in the Abag.h file that hadn't been written yet.

   I created a bag object in the bagtestmain.cpp file and tested each member function as it was written (only a couple errors during testing, but these were incrementally solved below:
   a. Illegal use of type as expression: the line that generated the error was **E[used-1]**. After taking some time to review, I noticed I was using the data type E instead of the name of the 'E' array **data**. Using **E** worked in the constructor, which threw me off a bit. Rewrote the line as **data[--used]** and replaced all instances of **E** outside of the constructor with **data**.
   b. Exception being thrown due to a return statement being above some code in a if/else block. This caused a memory address access violation (first iteration of 'if' could try to access location "-1" when the **data[--used]** snippet occurred). Moving the return statement below the logic as expected fixed this exception.

2. The following is the approach I took when writing each of the member functions:
   a. Constructors: Looking at bagtestmain.cpp, the bag class needed only one variable to perform every function: size, implement a default constructor and one that takes a size variable of size_t (tried int and didn't work, used size_t from cpp file)
   b. Destructor: default destructor is fine, deleting the array itself, but if I can use smart pointers, this portion can be left alone. (4/2/2023: I was able to make this array a smart pointer)
   c. Additem: check to see if there is room (used vs capacity), if there is, add the item at [used] location, then increment used.
   d. Remove: compare parameter against items within bag using iteration, if params match item, move each item after this one back one space via iteration, then decrement used.
   e. removeTop: as long as a value exists, decrement used (ensure this only happens if used is not equal to 0)
   f. Find: same idea as remove,but leave used variable alone; show the data at the location via parameters.
   g. inspectTop: sane idea as remove top, but show data at top, leave used variable alone; show via params.
   h. emptyBag: just like stack ADT from book, set used = 0
   i. Size: return used
   j. bagCapacity: return bagCapacity.

*Ask Mr. Christian:*

- I wrote code within the .cpp driver to test the Bag ADT's header/implementation file. Should I leave this there as evidence of my work, or should they be copied within the "approach" doc file?
- The bagtestmain.cpp file has a comment saying to "Demonstrate any Bag functions that were not used/demonstrated in the implementation of BDictionary/Abag using kv pairs, I haven't worked on dictionary yet, but I see the bag methods "inspectTop" and "bagCapacity" are not within the .cpp driver. Should I add them here?
- I reviewed smartPointers a few times, I see the opportunity for them in a few places, including the array within the ABag.h file. I am not asking for how to complete this, but I wanted to ask if this is a feasible use of the "shared ptr" smart pointer. If I am thinking in the entirely wrong way I just wanted to confirm/deny that, as the few times I tried using it, the results were not successful.

3/28/2023:

1. Tried wrestling with smart pointers again, time to make a note to review these for a third time and come back later.
2. Wrote comments on ABag file to explain the code.

3/29/2023

1. Reviewed textbook and notes on dictionaries.
2. Started writing the methods for BDictionary ADT.

3/30/2023:

1. Approach for writing dictionary methods:
    a. Get a dictionary to be able to just be created via constructor, leave members commented out for now.
2. Finished writing the methods for the BDictionary.
    a. Program had a few syntax errors, but was quickly resolved and was able to compile.

3. My targets for how to write each member function:
    a. Constructor: I know from .cpp file that the constructor can take a variable of size, and from the comments from instructor, I know that I need a pointer to a bag where dictionary is assigned to it. Make it take a KVpair object with a size parameter. Put a smart pointer in if able (4/2/2023: smart pointer implemented)
    b. Destructor: blank default destructor is fine if smart pointer works, otherwise delete the dictionary object here.
    c. Clear: call the emptyBag method from the bag class with pointer to member access.
    d. Insert: Call the addItem method from the bag class with pointer to member access, kvpair as params(4/2/2023: smart pointer implemented)

    e. Remove: Call the remove method from the bag class with pointer to member access,  kvpair as params (4/2/2023: smart pointer implemented)
    f. RemoveAny: call the removeTop method to remove an item from any bag, use kvpair as params
    g. Find: call the find method from the bag class with pointer to member access, kvpair as params (4/2/2023: smart pointer implemented)
    h. Size: call size method from bag class with pointer to method access

4. Finished writing the methods for the BDictionary.
    a. Program had a few syntax errors, but was quickly resolved and was able to compile.
    b. The values within the console were either missing or nonsense.

5. Many opportunities for smart pointers present in the BDictionary methods I have written; I will do that last since this portion is worth the least amount of points according to the rubric.
6. I was able to fix the nonsense values from 4b: reviewing the KVpair.h file, I never utilized the member access function for value(). Adding these to each of the remove and searching functions for the dictionary class seems to have fixed the issue, but I want to test this a bit more before I just assume it's good
    a. Used if clause to ensure the E& value of these functions is only assigned if the item is able to be located and/or removed.
7. Commented the entire BDictionary.h file to explain the code.

3/31/2023:

1. Begin smart pointer implementation on dictionary file, this seems to have the most to gain from smart pointers.

4/2/2023:

1. Encountered a few syntax errors when putting in smart pointers, but this was mostly due to forgetting/mixing up "unique_ptr" and "make_unique" declarations.
2. Was also able to implement smart pointers in ABag.h by using the array brackets (learned this by googling how to declare a unique pointer to an array)
3. Added functions to bagtestmain.cpp to test bag methods that were not obvious from given code (added functions to test "addItem", "+=", "bagCapacity", and "inspectTop")

4/3/2023:

1. Reviewed code to ensure output is correct in the console and free of errors.
2. Put project on flash drive to test ability to be compiled on different computer (success)
3. Reviewed comments for leftover issues to be resolved
4. Reviewed assignment instructions to ensure all action items are completed.