**CSCI 232 - Out Lab 2 (100 pts max)**

**Due: 11pm, October 2, 2020**

**Overview:**

In this lab, you will write a program called BinPacker; it should take a bin <capacity> value and a <filename> as command-line arguments. Input files contain a list of item sizes (all values will be positive integers), e.g.:

28

42

12

5

30

The objective of this lab is to test various strategies for the bin packing problem, see the following for more information about the problem:

<https://en.wikipedia.org/wiki/Bin_packing_problem>

Briefly, the problem is, given a list of items of various sizes (the values in file), find the fewest number of bins that hold all of the items, assuming each bin can hold no more that

<capacity> values.

Your program should run from the command line:

java BinPacker 50 input.txt

**Notes:**

* You should implement the following strategies for solving the problem:

First-fit

The algorithm processes the items in the order they arrive. It attempts to place each item in the first available bin that can accommodate it. If no bin is found, it opens a new bin and puts the item into the new bin.

BestFitDecreasing

You put the next item into the bin that has the least remaining space among bins capable of storing the item. Use a binary search tree-based symbol table for this; create a method called **bestfit(x**) that takes an item size x and returns the bin whose remaining capacity is at least x and is closest to x. If no bins can hold x, create a new bin. *Note: there could be multiple bins with the same remaining capacity; think about how to handle this!*

WorstFitDecreasing

You put the next item into the bin that has the most remaining space among bins capable of storing the item. Again, use a binary search tree-based symbol table for this; create a method called **worstfit(x**) that takes an item size x and returns the bin whose remaining capacity is the greatest and can hold x. If no bins can hold x, create a new bin.

* If an item size exceeds the given bin capacity, print out a note saying that an item with greater weight than the bin capacity was encountered.
* You may use a built-in BST class, or one from the textbook.

**Output:**

Your program should print out the item placement achieved by each strategy. One average, which strategy works best. Can you come up with ‘good’ cases for each strategy, where that strategy finds the best solution but the others do not?

*Some points will be awarded for nicely-formatted output.*

**Submission:**

Submit using gradescope prior to the due date/time (you may also submit in-person directly to your TA during lab). Submit your:

1. .java files
2. an input file that you’ve created to test the program
3. the resulting output and your discussion (can be one document).

*Only one submission per group is required BUT you must put ALL group member names in the comments of your program.*

**Bonus Problem: Implement Your Own BST (+ 5 bonus pts)**

Implement your own binary search tree class and necessary methods (it does not need to be balanced).