Assignment 6

Due Sunday, August 18th, 2013

This assignment is worth 6% of your grade.

Submit your code on cuLearn in ONE file called a6.c. Your code (that is a6.c) must compile, otherwise the assignment will be graded with mark 0. You may use any of the libraries that we have seen in class:
`stdio.h, math.h, string.h, stdlib.h`

To solve this assignment you will need to know (maybe dynamic) arrays and linked lists. So if you want to start working on this before Tuesday morning, read Chapter 13.

Your program should implement a data structure call *bucket-chains*, or *BC*, for short, as detailed in what follows.

You should write a program that does the following:
****

1) Opens the file called "input.txt". I will provide a sample of this file on cuLearn. You should put it in the same folder on your computer where a6.c resides. This file contains real numbers (type `double` in C) in the range [0, 500), that is, each number is greater or equal to zero and strictly smaller than 500. You program should first insert into your BC data structure all the numbers that it finds in input.txt file. What I mean by insert, will become clear shortly.

Your BC data structure should have 20 buckets, each implemented as a linked list (where the numbers in each linked list appear from the smallest to the largest. In other words each bucket is an ordered linked list). The numbers in the range [0,25) go to the (linked list of the) first bucket, numbers in the range [25, 50) go to the (linked list of the) 2nd backet, ..., the numbers in the range [475, 500) go into the last, that is 20th, (linked list) backet.

(To implement this think of having an array of pointers pointing to the beginning/head of each linked list – as depicted in the figures later on).

2). Your program should then display a MENU to the user that looks like this:

```
Choose one of the following options (by entering the number corresponding to
the desired task):

1. Test if a given number is in BC data structure
2. Insert a number into BC data structure.
3. Delete a number from BC data structure.
4. Display a content of a given bucket.
5. Print buckets in order (which will result in sorted list of numbers)
6. Exit the program.
```

After your program completes the task chosen by the user, it should ask the user to press zero in oder to go back to the menu, unless the user chose 6 previously, in which case the program terminates.

Write a function for each of the tasks 1 to 5 above. (Those functions can call any other functions that you created). Global variables are not allowed.

I will explain now what each of the above tasks should do. At the end of all the explanations, I will draw some figures which will make it much easier to understand what is required. So you may want to look at the figures before reading on.

Here is what each task above should do:

1. (Task 1: search) Your program should ask the user to enter a number to be searched for in the [1,500) range. It should test if that number is already in you data structure and inform the user of the result of that search.

2. (Task 2: insert) Your program should ask the user to enter a number, in the [1,500) range, to be inserted. It should then insert that number in the linked list of the appropriate bucket, if the number is not already there. The number should be inserted in the appropriate spot of the linked list such that the linked list remains sorted after the insertion.

3. (Task 3: delete). Your program should ask the user to enter a number, in the [1,500) range, to be deleted. It should then go to the appropriate bucket and delete the numer in questions from the corresponding linked list, while keeping the linked list sorted. If the number is not present, your program should alert the user to that fact by printing "Nothing to delete".

4. (Task 4: print bucket). Your program should ask the user to enter an integer, in the [1,20] range, for the bucket number. It should then print numbers in the linked list corresponding to the chosen bucket.

5. (Task 4: print all buckets, i.e., sorted numbers). Your program should print buckets one by one from the first to the last (i.e. 20th) by printing each of its linked lists from the first element of the list to the last. This should result in printing the numbers in order, from the smallest to to largest.

You can test your program on the file input.txt that I provided. However, the TA will test your program on a different input.txt file. Your program should thus work for any file as long as that file is called `input.txt` and contains numbers in range [0,500). You can assume that the numbers are separated by either a space or a newline.

For bouns points: Above, the number of buckets is fixed to 20. Instead, for bonus points, your program should do the following, first go through the input.txt file to find the maximum number in the file. Let x the first integer greater or equal to that number that is divisible by 25. Your program should create BC data structure with $x/25$ buckets. Then the numbers in the range $[0, x/25)$ would go in the first bucket, the numbers in the rage $[x/25, 2x/25)$ should go in the second bucket, and more generally the numbers in the range $[(i-1)x, ix)$ should go in the i-th bucket. It should also instruct the user to enter numbers in the range $[0, x)$. Also there would be no assumption on the maximum number in the input.txt file (above that max was assumed to be 500).

Figures on the next page

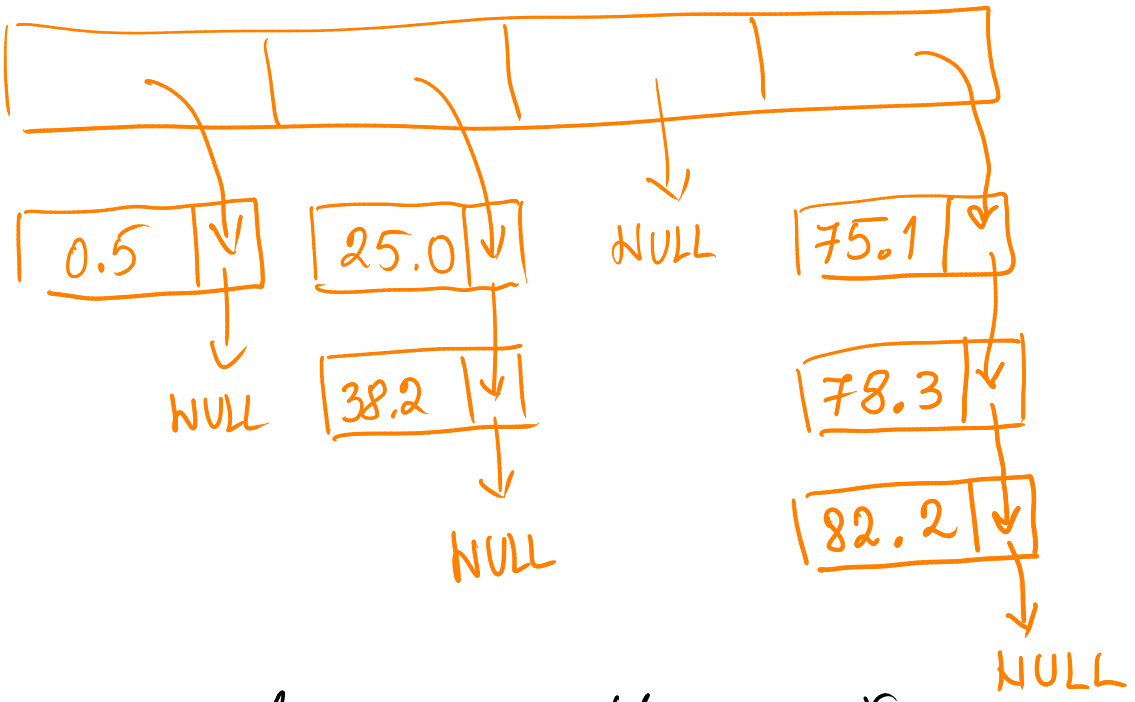
btw. you will be graded on all programming aspects (as implicit always) including and not limited to: code readibility and conciseness, efficiency of the solution etc ....

Example with 4 buckets and numbers in the range [0, 100) where numbers from [0, 25) go to bucket 1, numbers from [25, 50) go to bucket 2...

Let's say input.txt had following numbers:

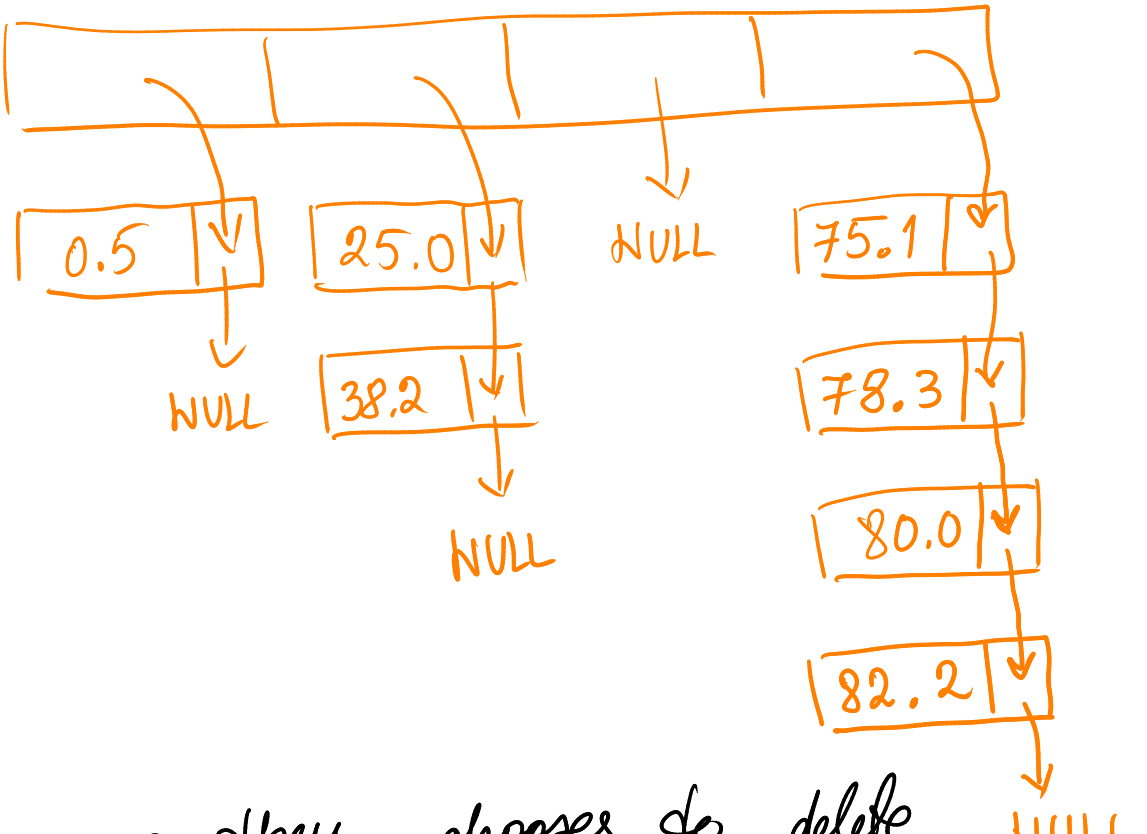
78.3 0.5 38.2 82.2 25.0 75.1

then after step 1 of your program BS data structure would look like this:



In the second part, assume the user chose to print 4th bucket. Your program would print 75.1, 78.3, 82.2

- If user asked to print 3rd bucket your program would print nothing.
- If the user then chooses to insert 80.0, the BC data structure would look like this:



If the user then chooses to delete 78.3 the BC data structure would look like this: