323.33 Spring 2020 Mock Final - 2 Name:___Andres Quintero

In the first mock exam, you wrote a specs for the implementation of the selection-sort algorithm, using an array as the main data structure. In this mock exam, instead of using an array, you will use a linked list (with a dummy node) to store the data. The sorting method used is the same selection-sort. Instead of loading the data into an array, your program will load the data into a linked list (read a data from the input file, get a newNode for the data, then, inserts newNode in the front of the list, after the dummy node). Sorting process works the same way -- conducted in iterations. After sorting, it outputs the sorted data in the linked list to a output file.

You are to write the specs for selection-sort (**in ascending order**), using linked list: 1. Programming language (your choice):

C++

2. Input specification. (Need to be more precise and clear!)

inFile (use argv[1]) – a text file with intergers separated by spaces

3. Output specification, write whatever you like to see in the out files. (Need to be more precise and clear!)

outFile(use argv[2]) – a text file with integers on each line sorted in ascending order

3. Write the data structure (using linked list) // similar to the linked list data structures in all your projects).

```
Class Node

      -  data (int)
      -  prev (Node*)
      -  next (Node*)
      -  Node(int d) data = d
```

```
Class SelectionSort

      -  head = new Node(-9999) (Node*)
      -  int N

         createList(inFile)
         0- inFile <- given
         1- data <- read from inFile
         2- newNode <- create with data
         3- N++
         4- Create temp (node *)
         5- head.next <- newNode
         6- newNode.prev <- head
         7- newNode.next <- temp
         8- repeat 1 - 8 until inFile.eof()
            selectionSorting() // Not working
            0- position <- head.next
            1- lowestFound <- position
            2- findingNode <- position.next
            3- if(findingNode.data < lowestFound.data)
               lowestFound <- findingNode
            4- findingNode <- findingNode.next
            5- repeat 3-5 while findingNode.next != NULL;
            6- if(lowestFound.data < position.data)
                  temp <- create
                  temp.next <- lowestFound.next
                  temp.prev <- lowestFound.prev
                  lowestFound.next <- position.next
                  lowestFound.prev <- positin.prev
                  position.next <- temp.next
                  position.prev <- temp.prev;
            7- position < position.next
            8- repeat 1 - 8 while position.next.next != NULL



            pritnList(outFile)

            0- outFile <- given
            1- spot <- head.next
            2- spot.data -> outFile
            3- spot <- spot.next
            4- repeat 2-4 while spot.next != NULL
```

4. Write the algorithm steps for main (...). (Write other methods on the next page.)

```
-0 Open inFile, outFile <- given

-1 createList(inFile)

-2 selectionSOrting()

-3 printList(outFile)

-4 close all files
```

6. **Follow your specs to implement your Selection sort, in ascending order.**

III. Submission: submit the following 3 separate files in the same email:
- The essay questions and answers (1 to 5) in any option you choose in the above. - Soft copy of programming question
- Hard copy (cover page, source code, and all outputs)
