**CMSC 3621**

**Lab 4**

Complete the implementation of a linked selection sort in the following two methods.

1)

template <class Record>

void Sortable\_list<Record>::min\_node(Node<Record>\* start\_node,

Node<Record>\* start\_node\_prev, Node<Record>\* &min\_node,

Node<Record>\* &min\_node\_prev)

2)

template <class Record>

void Sortable\_list<Record>::swap(Node<Record>\* start\_node,

Node<Record>\* start\_node\_prev, Node<Record>\* min\_node,

Node<Record>\* min\_node\_prev)

Requirements:

1. Your implementation must be based on "Linked selection sort" under Assignment 4 in the URL <https://cs2.uco.edu/~gqian/cmsc3621/assignments>. The algorithm has been implemented in selection\_sort, which uses two helper functions, including min\_node (incomplete) and swap (incomplete). Make sure that you read the code and comments of the three methods, selection\_sort, min\_node and swap, to understand the implementation.
2. You are required to complete the helper function min\_node in **Sortable\_list.h**. Specifically, you are asked to complete the body of the while loop. The code you add should identify the node with the smallest value (min\_node) and its predecessor node (min\_node\_prev).
3. You are required to complete the helper function swap in **Sortable\_list.h**. Specifically, you are asked to implement statements of pointer manipulation so that the two nodes (start\_node and min\_node) are swapped in the linked list. Note that swap considers two cases for swapping the two nodes as follows:
4. Case 1: start\_node and min\_node\_prev are not the same node in the linked list.

... -> start\_node\_prev -> start\_node -> ... -> min\_node\_prev -> min\_node -> ...

1. Case 2: start\_node and min\_node\_prev are the same node in the linked list.

... -> start\_node\_prev -> start\_node / min\_node\_prev -> min\_node-> ...

1. When you finish the two methods in Sortable\_list.h, you can use option 1 in the user interface to import one of the provided data file ("dl.txt", "l2k.txt", and "l200k.txt") and then use option 2 to sort the imported list using selection sort. Note that it may take up to four minutes for the algorithm to sort l200k.txt. When sorting is done, you can use option p to display the original list and use option s to display the sorted list.
2. Create a subdirectory named "lab4" under your server account on cs.uco.edu. Upload all your source code files, "makefile", and the three data files ("dl.txt", "l2k.txt", and "l200k.txt") to your server account under "lab4". The instructor should be able to use "make" to compile and test-run your program.
3. Use nano or vi to create a file named "answers" under "lab4". Add your answers to the following questions into "answers".

Review selection\_sort in Sortable\_list.h and then answer the three questions below:

* What is the purpose of defining dummy with its next being this->head on line 26?
* On lines 45-46, the following code is used to set current (and prev) to the next node in the list for the subsequent iteration of the while loop:

prev = prev->next;

current = prev->next;

Why don’t we use the following code, which seems to accomplish the same purpose and appears to be more intuitive?

prev = current;

current = current->next;

* On line 49, why is this->head reset to dummy\_ptr->next, which seems to be unnecessary as dummy\_ptr->next has already been set to this->head on line 26 where dummy is defined?

Submission:

The following files must be available under "lab4" in your server account on cs.uco.edu:

1. The source code files, "makefile", and the three data files ("dl.txt", "l2k.txt", and "l200k.txt") for "Linked selection sort" with your completed implementation of min\_node and swap
2. The compiled, executable program "sel"
3. The file "answers" with your answers to the questions in step 6)