Student ID: Student Name:

Course: Data Structures (CSE CS203A)

Assignment III: Linked List Selection Sort

Due date: 2025.10.20 10:00:00

Important Notice: You must print this assignment (Student Worksheet Companion) and **write your answers by hand with a pen**.

Goal

Practice representing data using a **linked list**, apply the **selection sort** algorithm step by step, and analyze the **differences between array and linked list** implementations in terms of traversal, exchange cost, and computational overhead.

Given Data

Input integers: 60, 24, 15, 42, 20, 11, 90, 8

Background

Linked List: A linked list is a sequence of nodes stored in **non-contiguously memory**. Each node contains a **value field** and a **pointer field** that links to the next node. Traversal is sequential O(n), but **insertion** or **deletion** (given a node) is O(1).

Selection Sort Algorithm: Selection sort is an **in-place comparison-based** algorithm that divides the structure into **sorted** and **unsorted** portions. For each position i, it finds the minimum element in the remaining unsorted portion and swaps it with the element at position i.

Selection Sort Algorithm

Basic Idea: For each position i from 0 to n-2, find the minimum element in the unsorted suffix (from position i+1 to n-1) and swap it into position i.

Pseudocode:

```
SELECTION-SORT-LIST(head):
    i = head
    while i != NULL and i->next != NULL:
        min_node = i
        j = i->next
        while j != NILL:
            if j->val < min_node->val:
```

Student ID: Student Name:

```
            min_node = j
        j = j->next
    if min_node != i:
        swap i->val, min_node->val
    i = i->next
```

Tasks

Approach

Represent the integers in a singly linked list data structure and apply the selection sort algorithm step by step. In this pseudocode, only swap the values not modify the next pointer to complete the sorting algorithm

Analysis: Compare Array (Assignment II) vs Linked List in terms of exchanges, traversal, overhead, and visualization.

Required Answer Parts

A1. Draw the visual representations of the node in the singly linked list structure

A2. Populate the linked list with the given integers and add appropriate annotations (head pointer, next pointers and values)

A3. The first step is the demo step for you to complete the following three steps of the selection sort algorithm with detailed traces

A4. Discussion: Analyze and compare Array vs Linked List.

Deliverables

Print the **Student Worksheet Companion and submit** handwritten worksheet containing:

- Linked list structure drawings (A1)
- Populated linked list (A2)
- Step-by-step traces of the first 3 steps (A3)
- Comparative discussion (A4)

Evaluation (100 pts)

- Representations (A1): 5 pts
- Populated linked list (A2): 32 pts
- Linked list selection sort steps (A3): 25 pts
- Discussion quality (A4): 40 pts

Student ID: Student Name:

Course: Data Structures (CSE CS203A)

Assignment III: Linked List Selection Sort

Student Worksheet Companion

A1. Linked List Representation Drawing (5 pts)

a. (2 pts) Instructions: Draw a visual representation of a single node with next pointer that contains the initialized integer 10

b. (3 pts) Linked list representation with the given integers (Hint: For safety and clarity, include identifiable head and tail nodes)
Example: the input integers are (10, 20) and linked list representation will be [10 | ●] → [20 | ●] →

A2. Populate with Integers (32 pts; 2 pts for each)

Fill the given integers (60, 24, 15, 42, 20, 11, 90, 8) into the above structures.

Annotate:

Node #	Value	Next Pointer
1	[_____]	→ Node [_____]
2	[_____]	→ Node [_____]
3	[_____]	→ Node [_____]
4	[_____]	→ Node [_____]
5	[_____]	→ Node [_____]
6	[_____]	→ Node [_____]
7	[_____]	→ Node [_____]

Student ID: Student Name:

8	[_____]	→ [_____]

A3. Selection Sort – First Three Steps (45 pts; 15 pts for each step)

Step Trace Table (Linked list):

Step 1 is the example to help you to complete step 2 to 4.

Step 1 (i = head = 60): Traverse list to find minimum value 8 → call swap function Yes; swap (60, 8).

head → [8|●] → [24|●] → [15|●] → [42|●] → [20|●] → [11|●] → [90|●] → [60|NULL]

Step 2 (i = ____): Minimum value [____] → call swap function Yes / No; swap ([____], [____]).

head → [8|●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |NULL]

Step 3 (i = ____): Minimum value [____] → call swap function Yes / No; swap ([____], [____]).

head → [8|●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |NULL]

Step 4 (i = ____): Minimum value [____] → call swap function Yes / No; swap ([____], [____]).

head → [8|●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |●] → [____ |NULL]

Student ID: Student Name:

A4. Discussion (68 pts)

Guiding Questions:

- How many swaps/exchanges are performed?
- How expensive is traversal for arrays vs. linked lists?
- What memory/overhead differences do you see?
- Which representation is easier to visualize?
- Which would you choose for implementing selection sort and why?

Time complexity comparison (14 pts, 1pt for each)

Aspect / Operation	Array	Linked List	Explanation
Access Element	(1)	(2)	Array allows direct indexing; linked list needs traversal.
Find Minimum	(3)	(4)	Both must scan all remaining elements/nodes.
Swap Operation	(5)	(6)	In array, swap by indices; in linked list, swap node values.
Traversal Between Elements	(7)	(8)	Linked list traversal requires pointer navigation.
Overall Time Complexity (Selection Sort)	(9)	(10)	Both involve nested traversal to find minima; linked list adds traversal overhead.
Space Complexity	(11)	(12)	Both sorts are in-place if swapping values, not nodes.
Implementation Overhead	(13) Low or Moderate	(14) Low or Moderate	Linked list needs pointer operations and careful null checks.

Student ID: Student Name:

(1)		(2)	
(3)		(4)	
(5)		(6)	
(7)		(8)	
(9)		(10)	
(11)		(12)	
(13)		(14)	

Student ID: Student Name:

Characteristics (54 pts, 3 pts for each)

Aspect	Array	Linked List
Storage	(1)	(2)
Access	(3)	(4)
Extra Variables	(5)	(6)
Traversal	(7)	(8)
Overhead	(9)	(10)
Visualization	(11)	(12)
Swaps	(13)	(14)
Flexibility	(15)	(16)
Overall	(17)	(18)

(1)

(2)

(3)

(4)

(5)

(6)

(7)

(8)

(9)

Student ID: Student Name:

(10)

(11)

(12)

(13)

(14)

(15)

Student ID: Student Name:

(16)

(17)

(18)