|  |
| --- |
| **COSC 2436**  **Fall 2024 - Practice Exam 1**  Name: |

|  |  |  |  |
| --- | --- | --- | --- |
| **Section** | **Question #** | **Points Possible** | **Score** |
| Linked List | I | 12 |  |
|  | II | 12 |  |
|  | III | 8 |  |
|  | IV | 7 |  |
| Recursion | I | 12 |  |
|  | II | 10 |  |
| Sorting | I | 7 |  |
|  | II | 8 |  |
|  | III | 4 |  |
| Array | I | 8 |  |
| Time Complexity | I | 6 |  |
|  | II | 6 |  |
| **Total** | **12** | **100** |  |

**Linked Lists**

**1.** Write a function called **findMiddle()** which finds the middle of a linked list. **You may only use one for loop or while loop to do so.**

**struct node{**

**int value;**

**node \*next;**

**node(int v) : value(v), next(nullptr) {}**

**};**

**node\* findMiddle() {**

**}**

2. Write the function **node\* removeSmallest()** which removes the smallest value in a doubly linked list.

**node \*removeSmallest(12 ⇄ 26 ⇄ 4 ⇄ 2 ⇄ 78) => 12 ⇄ 26 ⇄ 4 ⇄ 78**

**struct node{**

**int value;**

**node \*next;**

**node(int v) : value(v), next(nullptr) {}**

**};**

**node \*removeSmallest(node\* head) {**

**//edge case**

**if (head == nullptr) return nullptr;**

**if (head→next == nullptr) {**

**delete head;**

**return nullptr;**

**}**

**//traverse**

**node\* cu = head;**

**node\* min = head;**

**while (cu != nullptr) {**

**if (cu→value < min→value) {**

**min = cu;**

**}**

**cu = cu->next;**

**}**

**//reroute**

**//min is at end**

**if (min→next == nullptr) {**

**min→prev = nullptr;**

**} else {**

**min→next→prev = min→prev;**

**}**

**//min is at start**

**if (min→prev == nullptr) {**

**head = min→next;**

**} else {**

**min→prev→next = min→next;**

**}**

**//set min to nullptr**

**min→next = nullptr;**

**min→prev = nullptr;**

**delete min;**

**return head;**

**}**

3. Write a function which calculates how many times a **sorted singly linked list** has been rotated around.

List: 4 → 5 → 1 → 2 → 3

Output: 2

**int numRotations(4 → 5 → 1 → 2 → 3) => 2**

**struct node{**

**int value;**

**node \*next;**

**node(int v) : value(v), next(nullptr) {}**

**};**

**int numRotations(node \*head){**

**}**

4. You are given **two** linked lists of equal size which first have even numbers, then odd numbers, merge them so that the resulting list is as follows : **even numbers in order - odd numbers in order**. Return the head of the resulting linked list.

**List 1: 2 → 6 → 10 → 3 → 7 → 9**

**List 2: 4 → 8 → 12 → 1 → 13 → 15 → 19**

**Output: 2 → 4 → 6 → 8 → 10 → 12 → 1 → 3 → 7 → 9 → 13 → 15 → 19**

**node \*mergeWithPattern(node\* head) {**

}

**Recursion**

1. Write a recursive function which returns the sum of factorials at **every 2nd index**. You have access to the **factorial(int n)** function.

**sumEverySecondFact(4, 1)**

Indexes: [1, 2, 3, 4]

Pick every 2nd index: 2 + 24 = 26

**Final Output: 26**

**void sumEverysecondFactorial(int n, int index){**

**}**

2. What is the output of **func(4)?**

**void func(int n){**

**if(n >= 1){**

**func(n - 1);**

**n--;**

**func(n - 1);**

**cout << n << " ";**

**}**

**}**

**Sorting**

1. Perform Bubble sort and Insertion sort on the following array. Show your steps.

{2, 5, 9, 11, 2, 4}

|  |
| --- |
| **Bubble:**  **2,5,9,11,2,4**  **2,5,9,2,11,4**  **2,5,9,2,4,11**  **2,5,2,9,4,11**  **2,5,2,4,9,11**  **2,2,5,4,9,11 2,2,4,5,9,11**  **Insertion:**  **2,5,9,11,2,4**  **2,5,9,11,2,4**  **2,5,9,11,2,4**  **2,5,9,11,2,4**  **2,2,5,9,11,4**  **2,2,4,5,9,11** |

**2.** Describe Each sorting algorithm and what it does. Identify Time Complexity in Big-O notation (**fastest and slowest**).

**Bubble:**

**Compares neighboring elements and swaps if i>i+1. Continues to do so until early sorted (bubble v2) or passes through the array n-1 time in worst case.  
  
Worst case: O(n^2)**

**best case: O(n)**

**Insertion:**

**Com**

**Arrays**

How do you initialize a Dynamic 2D array? Show code snippet.

**Int arr[arr[x]] = [[]]**

How would you fill said array? Show code snippet and explain.

**Time Complexity**

1. What is the time complexity of the following code? **What makes it that way?**

**for(int i = 1; i < n; ++i){**

**for(int j = 10; j < n; j \*= 2){**

**cout << “Easy Points!” << endl;**

**}**

**}**

2. What is the time complexity of the following code?

**void func(int n){**

**if(n < 0)**

**return;**

**else{**

**func(n-2);**

**int j = n \* 2;**

**cout << n;**

**func(n-3);**

**}**