MTH 4300: Algorithms, Computers and Programming II

Spring 2026

Section: SMWA

Midterm 1 (March 4th 2026)

Grading Section (For Instructor Use)

Question	Points	Score
Question 1	5	/ 5
Question 2	5	/ 5
Question 3	7	/ 7
Question 4	10	/ 10
Question 5	10	/ 10
Total	**37**	/ 37

Comments:

Question 1

What is the output of the following program? Trace through step by step, tracking the value of every variable and what each pointer points to after each statement.

```
#include <iostream>

void update(int*& p, int& r) {
    r += *p;
    p = &r;
}

int main() {
    int a = 4, b = 6, c = 10;
    int* ptr = &c;
    int& ref = a;

    std::cout << *ptr << " " << ref << std::endl;

    update(ptr, b);

    std::cout << *ptr << " " << ref << std::endl;
    std::cout << a << " " << b << " " << c << std::endl;

    ref += *ptr;
    *ptr = ref - c;

    std::cout << a << " " << b << " " << c << std::endl;
    std::cout << *ptr << " " << ref << std::endl;

    return 0;
}
```

Question 2

What is the output of the following program? For each recursive call, show the value of `i`, what is printed (if anything), and what value is returned.

```
#include <iostream>
#include <vector>

int f(std::vector<int>& v, int i) {
    if (i == (int)v.size() - 1) return v[i];
    int rest = f(v, i + 1);
    v[i] = rest - v[i];
    std::cout << v[i] << std::endl;
    return v[i] + rest;
}

int main() {
    std::vector<int> v = {2, 5, 3, 4};
    int r = f(v, 0);
    std::cout << r << std::endl;
    std::cout << "Vector: ";
    for (int i = 0; i < (int)v.size(); i++) {
        std::cout << v[i];
        if (i < (int)v.size() - 1) std::cout << " ";
    }
    std::cout << std::endl;
    return 0;
}
```

Question 3

The program below intends to build a linked list by appending values one at a time, print the result, then free all memory. It contains **2 bugs**. For each one, state the line, explain the problem, and write the fix.

```
#include <iostream>

struct Node {
    int data;
    Node* next;
};

void append(Node*& head, int value) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->next = nullptr;

    Node* current = head;
    while (current->next != nullptr) {
        current = current->next;
    }
    current->next = newNode;
}

void printList(Node* head) {
    while (head->next != nullptr) {
        std::cout << head->data << " -> ";
        head = head->next;
    }
    std::cout << "nullptr" << std::endl;
}

void deleteList(Node* head) {
    while (head != nullptr) {
        Node* next = head->next;
        delete head;
        head = next;
    }
}

int main() {
    Node* list = nullptr;
    append(list, 10);
    append(list, 20);
    append(list, 30);

    printList(list);
    deleteList(list);
    return 0;
}
```

Question 4

Write a recursive function with the following signature:

```
std::string everyOther(const std::string& s, int index)
```

Returns a new string containing only the characters of `s` at even positions (`index` 0, 2, 4, …). No loops allowed.

Examples:

```
everyOther("abcdef", 0) → "ace"
everyOther("hello", 0) → "hlo"
everyOther("x", 0) → "x"
everyOther("", 0) → ""
```

Part A: Pseudocode

Detail your solution using pseudocode

Part B: Implementation

Implement the function above in C++

Question 5

Given the following struct:

```
struct Node {
    int data;
    Node* next;
};
```

Write a function with the following signature:

```
Node* mergeSorted(Node* a, Node* b)
```

Both `a` and `b` are the heads of sorted singly linked lists in non-decreasing order. Return the head of a single merged sorted list built by relinking the existing nodes — do not allocate any new nodes.

Examples:

```
1 -> 3 -> 5 and 2 -> 4 -> 6 → 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> nullptr
1 -> 2 -> 3 and nullptr → 1 -> 2 -> 3 -> nullptr
nullptr and nullptr → nullptr
```

Part A: Pseudocode

Detail your solution using pseudocode

Part B: Implementation

Implement the function above in C++