MTH 4300: Algorithms, Computers and Programming II

Fall 2025

Section: STRA

Midterm 1 Solutions (October 9th, 2025)

Question 1 Solution (10 points)

Implement a function that processes text content based on different operation modes using enums.

```cpp
#include <iostream>
#include <string>
#include <sstream>

enum ProcessMode { FIND_SHORTEST_WORD, FIND_LONGEST_WORD };

std::string processContent(const std::string& content, ProcessMode mode) {
    std::istringstream iss(content);
    std::string word;
    std::string result;

    if (mode == FIND_SHORTEST_WORD) {
        int minLength = INT_MAX;
        while (iss >> word) {
            if (word.length() < minLength) {
                minLength = word.length();
                result = word;
            }
        }
    } else if (mode == FIND_LONGEST_WORD) {
        int maxLength = 0;
        while (iss >> word) {
            if (word.length() > maxLength) {
                maxLength = word.length();
                result = word;
            }
        }
    }

    return result;
}
```

Alternative Solution (without istringstream)

```cpp
#include <iostream>
#include <string>

enum ProcessMode { FIND_SHORTEST_WORD, FIND_LONGEST_WORD };

std::string processContent(const std::string& content, ProcessMode mode) {
    std::string word = "";
    std::string result = "";

    if (mode == FIND_SHORTEST_WORD) {
        int minLength = INT_MAX;

        for (int i = 0; i <= content.length(); i++) {
            if (i == content.length() || content[i] == ' ') {
                if (!word.empty()) {
                    if (word.length() < minLength) {
                        minLength = word.length();
                        result = word;
                    }
                    word = "";
                }
            } else {
                word += content[i];
            }
        }
    } else if (mode == FIND_LONGEST_WORD) {
        int maxLength = 0;

        for (int i = 0; i <= content.length(); i++) {
            if (i == content.length() || content[i] == ' ') {
```

```
                if (!word.empty()) {
                    if (word.length() > maxLength) {
                        maxLength = word.length();
                        result = word;
                    }
                    word = "";
                }
            } else {
                word += content[i];
            }
        }
    }

    return result;
}
```

Question 2 Solution (10 points)

Implement a recursive function to solve the Collatz Conjecture.

```cpp
#include <iostream>

int collatzSteps(int n) {
    if (n == 1) {
        return 0;
    }

    if (n % 2 == 0) {
        return 1 + collatzSteps(n / 2);
    } else {
        return 1 + collatzSteps(3 * n + 1);
    }
}
```

Question 3 Solution (10 points)

Compress a string using run-length encoding.

```
#include <iostream>
#include <string>

std::string runLengthEncode(const std::string& input) {
    if (input.empty()) {
        return "";
    }

    std::string result = "";
    char currentChar = input[0];
    int count = 1;

    for (int i = 1; i < input.length(); i++) {
        if (input[i] == currentChar) {
            count++;
        } else {
            result += currentChar + std::to_string(count);
            currentChar = input[i];
            count = 1;
        }
    }

    result += currentChar + std::to_string(count);
    return result;
}
```

Question 4 Solution (10 points)

Rotate a 1D array representing a square matrix 90 degrees clockwise.

```
#include <iostream>

void rotateMatrix90(int matrix[], int size) {
    // Create a temporary array to store the result
    int* temp = new int[size * size];

    // Copy rotated values to temporary array
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            // Element at (i,j) goes to (j, size-1-i)
            temp[j * size + (size - 1 - i)] = matrix[i * size + j];
        }
    }

    // Copy back to original array
    for (int i = 0; i < size * size; i++) {
        matrix[i] = temp[i];
    }

    delete[] temp;
}
```