MTH 4300: Algorithms, Computers and Programming II

Fall 2025

Section: STRA

Midterm 1 Practice Exam (Oct 2, 2025)

Instructions: Show all your work. For code output questions, trace through the execution step by step. For implementation questions, write complete, compilable C++ code.

Note: When submitting your solutions for this practice exam, please make sure to submit a PDF with handwritten responses.

Question 1

What is the output of the following C++ program? Show your work by tracing through the execution step by step.

```
#include <iostream>
#include <vector>
using namespace std;

void modifyArray(vector<int>& arr, int* multiplier) {
    for (int i = 0; i < arr.size(); i++) {
        if (arr[i] % 2 == 0) {
            arr[i] *= (*multiplier);
        } else {
            arr[i] += (*multiplier);
        }
    }
    (*multiplier)++;
}

int main() {
    vector<int> data = {3, 8, 5, 12, 7};
    int factor = 2;

    cout << "Initial: ";
    for (int val : data) {
        cout << val << " ";
    }
    cout << "factor=" << factor << endl;

    modifyArray(data, &factor);

    cout << "After first call: ";
    for (int val : data) {
        cout << val << " ";
    }
    cout << "factor=" << factor << endl;

    modifyArray(data, &factor);

    cout << "After second call: ";
    for (int val : data) {
        cout << val << " ";
    }
    cout << "factor=" << factor << endl;

    return 0;
}
```

Question 2

What is the output of the following C++ program? Show your work by tracing through each recursive call.

```
#include <iostream>
using namespace std;

int transform(int n, int depth) {
    cout << "Called transform(" << n << ", " << depth << ")" << endl;

    if (depth == 0 || n <= 1) {
        cout << "Base case reached, returning " << n << endl;
        return n;
    }

    if (n % 2 == 0) {
        int result = transform(n / 2, depth - 1) + transform(n / 2, depth - 1);
        cout << "Even case: returning " << result << endl;
        return result;
    } else {
        int result = transform(n - 1, depth - 1) + 1;
        cout << "Odd case: returning " << result << endl;
        return result;
    }
}

int main() {
    int finalResult = transform(6, 2);
    cout << "Final result: " << finalResult << endl;
    return 0;
}
```

Question 3

Write a function `int findSecondLargest(const vector<int>& arr)` that finds the second largest unique element in an array. If there is no second largest element (array has fewer than 2 unique elements), return -1.

Examples:

- Input: `{5, 2, 8, 2, 9, 1}` → Output: `8` (largest is 9, second largest is 8)
- Input: `{3, 3, 3}` → Output: `-1` (only one unique element)
- Input: `{7, 7, 5}` → Output: `5` (two unique elements: 7 and 5)

Requirements:

- Do not sort the array or use any sorting functions
- Handle duplicate values correctly
- Use only one pass through the array if possible
- Write a complete `main()` function that tests your function with at least 4 test cases including edge cases

Question 4

Write a function `void reverseSegments(int* arr, int size, int segmentSize)` that reverses every segment of the specified size in an array using only pointer arithmetic.

Example:

- Original array: `{1, 2, 3, 4, 5, 6, 7, 8, 9}`
- After `reverseSegments(arr, 9, 3)`: `{3, 2, 1, 6, 5, 4, 9, 8, 7}`
- Explanation: Segments [1,2,3], [4,5,6], [7,8,9] are each reversed

Requirements:

- Use only pointer arithmetic (no array indexing with `[]`)
- Handle cases where the last segment is smaller than segmentSize
- If segmentSize ≤ 1 or ≥ size, do nothing to the array
- Write helper functions if needed
- Write a complete `main()` function that demonstrates your function with different test cases
- Display the array before and after the operation

Hint: You may want to write a helper function to reverse a single segment between two pointers.

Question 5

Write a recursive function `bool canPartition(const vector<int>& arr, int index, int sum1, int sum2)` that determines if an array can be partitioned into two subsets with equal sums.

Examples:

- Input: `{1, 5, 11, 5}` → Output: `true` (can be partitioned as {1, 5, 5} and {11})
- Input: `{1, 2, 3, 5}` → Output: `false` (cannot be partitioned equally)

Requirements:

- The function must use recursion (no loops in the main logic)
- At each step, decide whether to include the current element in subset 1 or subset 2
- Base case: when index reaches the end of array, check if sum1 == sum2
- Write a wrapper function `bool canPartition(const vector<int>& arr)` that calls your recursive function
- Write a complete `main()` function that tests your function with at least 3 different test cases

Approach Hint:

- Start with index 0, sum1 = 0, sum2 = 0
- For each element, try adding it to subset 1 OR subset 2
- Use logical OR to combine the results of both possibilities

Question 6

Write a function `vector<int> spiralTraversal(const vector<vector<int>>& matrix)` that traverses a 2D matrix in spiral order (clockwise from outside to inside) and returns the elements as a 1D vector.

Example:

- Input matrix:

```
1  2  3  4
5  6  7  8
9  10 11 12
```

- Output: `{1, 2, 3, 4, 8, 12, 11, 10, 9, 5, 6, 7}`
- Traversal path: right → down → left → up → right → down

Requirements:

- Handle rectangular matrices (not just square)
- Handle edge cases: empty matrix, single row, single column
- Use boundary variables to track the current "ring" being traversed
- Write a complete `main()` function that demonstrates your function with different matrix sizes
- Display both the input matrix and the spiral traversal result

Approach Hint:

- Use four boundary variables: top, bottom, left, right
- Traverse: left to right on top row, top to bottom on right column, right to left on bottom row, bottom to top on left column
- After each direction, update the corresponding boundary