

MTH 4300: Algorithms, Computers and Programming II

Fall 2025

Section: STRA

Midterm 2 Practice Exam 1

Question 1: Object-Oriented Programming

You are designing a racing game where players can drive different types of vehicles. Design and implement the following:

Base Class **Vehicle**:

- Private members: `driverId` (int), `baseSpeed` (double), `fuel` (double)
- Constructor that initializes all members
- Virtual destructor
- Public getter methods for all members
- Virtual method `double getMaxSpeed() const` that returns the base speed
- Method `bool consumeFuel(double amount)` that reduces fuel if enough is available, returns success
- Pure virtual method `void displayStats() const = 0`
- Pure virtual method `std::string getVehicleType() const = 0`

Derived Class **SportsCar**:

- Additional private members: `turboBoost` (double), `aerodynamics` (double)
- Constructor that properly initializes base class
- Override `getMaxSpeed()` to return `baseSpeed * (1.0 + turboBoost + aerodynamics * 0.1)`
- Implement `displayStats()` to print all vehicle stats including turbo and aerodynamics
- Implement `getVehicleType()` to return "SPORTS"

Derived Class **Truck**:

- Additional private members: `cargoWeight` (double), `durability` (double)
- Constructor that properly initializes base class
- Override `getMaxSpeed()` to return `baseSpeed * (1.0 - cargoWeight * 0.01)` (heavier cargo = slower)
- Implement `displayStats()` to print all vehicle stats including cargo weight and durability
- Implement `getVehicleType()` to return "TRUCK"

Function `simulateRace`: Write a function that takes a vector of `Vehicle*` pointers and a race distance (double):

- Calculate race time for each vehicle using: `time = distance / maxSpeed`
- Each vehicle consumes fuel equal to `distance * 0.1`
- Print race results showing driver ID, vehicle type, max speed, and finish time
- Return the driver ID of the winner (fastest time)

Question 2: Linked Lists

Given the following singly linked list node structure:

```
struct Node {  
    int data;  
    Node* next;  
    Node(int data) : data(data), next(nullptr) {}  
};
```

Write a function `Node* deleteAll(Node* head, int item)` that deletes all occurrences of a specific item.

input: `1 -> 2 -> 2 -> 3 -> nullptr`, `2`

output: `1 -> 3 -> nullptr`

Question 3: Stacks

Suppose that we have the following encoding rule: $k[\text{encoding_string}]$, where `encoded_string` inside the square bracket is being repeated exactly k times. Assume that $k > 0$.

Assume that the function prototype is `std::string decode_string(std::string s);`

input: `s = "3[a]2[bc]"`
output: `"aaabcbc"`

input: `s = "3[a2[c]]"`
output: `"accaccacc"`