WEEK 2 TASK

#include <iostream>

#include <unordered\_map>

// Initialize data structures for tracking available tickets, passengers, and earnings

std::unordered\_map<std::string, int> available\_tickets = {

{"09:00\_up", 6 \* 80},

{"11:00\_up", 6 \* 80},

{"13:00\_up", 6 \* 80},

{"15:00\_up", 6 \* 80},

{"10:00\_down", 6 \* 80},

{"12:00\_down", 6 \* 80},

{"14:00\_down", 6 \* 80},

{"16:00\_down", 6 \* 80 + 2 \* 80} // Extra coaches on the last train down

};

std::unordered\_map<std::string, int> passengers\_count = {

{"09:00\_up", 0},

{"11:00\_up", 0},

{"13:00\_up", 0},

{"15:00\_up", 0},

{"10:00\_down", 0},

{"12:00\_down", 0},

{"14:00\_down", 0},

{"16:00\_down", 0}

};

std::unordered\_map<std::string, int> earnings = passengers\_count;

// Function prototypes

void displayScreen();

std::string purchaseTickets(std::string journey, int num\_tickets);

void endOfDaySummary();

// Task 1 - Start of the day

void displayScreen() {

std::cout << "Train Schedule & Available Tickets" << std::endl;

for (const auto &ticket : available\_tickets) {

if (ticket.second > 0) {

std::cout << ticket.first << ": " << ticket.second << " tickets available" << std::endl;

} else {

std::cout << ticket.first << ": Closed" << std::endl;

}

}

}

// Task 2 - Purchasing tickets

std::string purchaseTickets(std::string journey, int num\_tickets) {

if (available\_tickets[journey] >= num\_tickets) {

available\_tickets[journey] -= num\_tickets;

int price = num\_tickets \* 25;

if (num\_tickets >= 10 && num\_tickets <= 80) {

int free\_tickets = num\_tickets / 10;

price -= free\_tickets \* 25;

}

earnings[journey] += price;

passengers\_count[journey] += num\_tickets;

return "Tickets purchased successfully.";

} else {

return "Not enough tickets available for this journey.";

}

}

// Task 3 - End of the day

void endOfDaySummary() {

int total\_passengers = 0;

int total\_earnings = 0;

std::cout << "\nEnd of the day summary:" << std::endl;

for (const auto &passenger : passengers\_count) {

std::cout << passenger.first << ": " << passenger.second << " passengers, Total earnings: $" << earnings[passenger.first] << std::endl;

total\_passengers += passenger.second;

total\_earnings += earnings[passenger.first];

}

std::cout << "\nTotal Passengers for the day: " << total\_passengers << std::endl;

std::cout << "Total Earnings for the day: $" << total\_earnings << std::endl;

std::string max\_passengers\_journey;

int max\_passengers = 0;

for (const auto &passenger : passengers\_count) {

if (passenger.second > max\_passengers) {

max\_passengers = passenger.second;

max\_passengers\_journey = passenger.first;

}

}

std::cout << "Journey with the most passengers: " << max\_passengers\_journey << " - " << max\_passengers << " passengers" << std::endl;

}

int main() {

displayScreen();

// Test purchasing tickets

std::cout << purchaseTickets("09:00\_up", 5) << std::endl;

std::cout << purchaseTickets("09:00\_up", 15) << std::endl;

std::cout << purchaseTickets("09:00\_up", 25) << std::endl;

// Test end of the day summary

endOfDaySummary();

return 0;

}