**2.Implement job sequencing with deadlines using a greedy method.**

#include <iostream>

#include <algorithm> // For std::sort

#include <string> // For string

using namespace std;

class Job {

public:

string id; // Job ID

int deadline; // Deadline of the job

int profit; // Profit of the job

// Default constructor

Job() {

id = "";

deadline = 0;

profit = 0;

}

// Parameterized constructor

Job(string id, int deadline, int profit) {

this->id = id;

this->deadline = deadline;

this->profit = profit;

}

};

// Comparator function for sorting jobs by profit in descending order

bool compareProfit(const Job &a, const Job &b) {

return a.profit > b.profit; // For descending order

}

void printJobSequence(Job jobs[], int n) {

// Sort jobs based on profit

sort(jobs, jobs + n, compareProfit);

// Find the maximum deadline

int maxDeadline = 0;

for (int i = 0; i < n; i++) {

if (jobs[i].deadline > maxDeadline) {

maxDeadline = jobs[i].deadline;

}

}

// Result array to store the job sequence and a slot tracker

string result[maxDeadline]; // To store job IDs

bool slot[maxDeadline]; // To track filled slots

// Initialize result with 'X' and slots as false

for (int i = 0; i < maxDeadline; i++) {

result[i] = "X";

slot[i] = false;

}

// Assign jobs to slots

for (int i = 0; i < n; i++) {

// Find a free slot for this job

for (int j = min(maxDeadline - 1, jobs[i].deadline - 1); j >= 0; j--) {

if (!slot[j]) { // Slot is free

result[j] = jobs[i].id; // Assign job id

slot[j] = true; // Mark this slot as filled

break;

}

}

}

// Print the job sequence

cout << "Job Sequence: ";

for (int i = 0; i < maxDeadline; i++) {

if (result[i] != "X") {

cout << result[i] << " ";

}

}

cout << endl;

// Calculate total profit based on the slots filled

int totalProfit = 0;

for (int i = 0; i < maxDeadline; i++) {

if (result[i] != "X") { // If this slot is filled

// Find the job corresponding to this ID and add its profit

for (int j = 0; j < n; j++) {

if (jobs[j].id == result[i]) {

totalProfit += jobs[j].profit; // Add profit of the job

break; // Exit loop once found

}

}

}

}

cout << "Total Profit: " << totalProfit << endl;

}

int main() {

int n;

cout << "Enter number of jobs: ";

cin >> n;

Job jobs[n]; // Declare a static array of jobs

cout << "Enter jobs in the format (id deadline profit):" << endl;

for (int i = 0; i < n; i++) {

string id;

int deadline, profit;

cin >> id >> deadline >> profit;

jobs[i] = Job(id, deadline, profit); // Initialize job object

}

printJobSequence(jobs, n); // Call function to print job sequence and total profit

return 0; // End of program

}