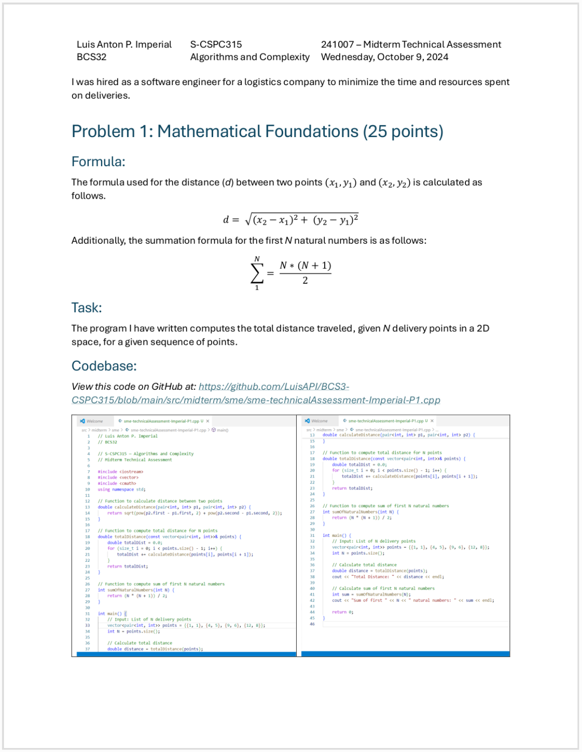
Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	Luis Anton Imperial
Assignment title:	Midterm Examination Turnitin Assessment
Submission title:	sme-technicalAssessment-Imperial.pdf
File name:	sme-technicalAssessment-Imperial.pdf
File size:	882.29K
Page count:	6
Word count:	610
Character count:	3,155
Submission date:	13-Oct-2024 02:40PM (UTC+0800)
Submission ID:	2483504627



Luis Anton P. Imperial S-CSPC315 241007 – Midterm Technical Assessment
BCS32 Algorithms and Complexity Wednesday, October 9, 2024

I was hired as a software engineer for a logistics company to minimize the time and resources spent on deliveries.

Problem 1: Mathematical Foundations (25 points)

Formula:

The formula used for the distance (d) between two points (x_1, y_1) and (x_2, y_2) is calculated as follows.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Additionally, the summation formula for the first N natural numbers is as follows:

$$\sum_{1}^{N} = \frac{N * (N + 1)}{2}$$

Task:

The program I have written computes the total distance traveled, given N delivery points in a 2D space, for a given sequence of points.

Codebase:

View this code on GitHub at: https://github.com/LuisAPI/BCS3-CSPC315/blob/main/src/midterm/sme/sme-technicalAssessment-Imperial-P1.cpp

Welcome
sme-technicalAssessment-Imperial-P1.cpp U
src > midterm > sme > sme-technicalAssessment-Imperial-P1.cpp > main()

```cpp
// Luis Anton P. Imperial
// BCS32

// S-CSPC315 – Algorithms and Complexity
// Midterm Technical Assessment

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

// Function to calculate distance between two points
double calculateDistance(pair<int, int> p1, pair<int, int> p2) {
    return sqrt(pow(p2.first - p1.first, 2) + pow(p2.second - p1.second, 2));
}

// Function to compute total distance for N points
double totalDistance(const vector<pair<int, int>>& points) {
    double totalDist = 0.0;
    for (size_t i = 0; i < points.size() - 1; i++) {
        totalDist += calculateDistance(points[i], points[i + 1]);
    }
    return totalDist;
}

// Function to compute sum of first N natural numbers
int sumOfNaturalNumbers(int N) {
    return (N * (N + 1)) / 2;
}

int main() {
    // Input: List of N delivery points
    vector<pair<int, int>> points = {{1, 1}, {4, 5}, {9, 6}, {12, 8}};
    int N = points.size();

    // Calculate total distance
    double distance = totalDistance(points);
```

Welcome
sme-technicalAssessment-Imperial-P1.cpp U
src > midterm > sme > sme-technicalAssessment-Imperial-P1.cpp > ...

```cpp
double calculateDistance(pair<int, int> p1, pair<int, int> p2) {
}

// Function to compute total distance for N points
double totalDistance(const vector<pair<int, int>>& points) {
    double totalDist = 0.0;
    for (size_t i = 0; i < points.size() - 1; i++) {
        totalDist += calculateDistance(points[i], points[i + 1]);
    }
    return totalDist;
}

// Function to compute sum of first N natural numbers
int sumOfNaturalNumbers(int N) {
    return (N * (N + 1)) / 2;
}

int main() {
    // Input: List of N delivery points
    vector<pair<int, int>> points = {{1, 1}, {4, 5}, {9, 6}, {12, 8}};
    int N = points.size();

    // Calculate total distance
    double distance = totalDistance(points);
    cout << "Total Distance: " << distance << endl;

    // Calculate sum of first N natural numbers
    int sum = sumOfNaturalNumbers(N);
    cout << "Sum of first " << N << " natural numbers: " << sum << endl;

    return 0;
}
```