XJTLU Entrepreneur College (Taicang) Cover Sheet

Module code and Title	**DTS102TC Programming with C++**
School Title	**School of Artificial Intelligence and Advanced Computing**
Assignment Title	**Coursework 1 (Assignment)**
Submission Deadline	**5 pm China time (UTC+8 Beijing) on** Fri. 28th. Nov. 2025
Final Word Count	**NA**
If you agree to let the university use your work anonymously for teaching and learning purposes, please type **"yes"** here.	**yes**

I certify that I have read and understood the University's Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy I certify that:

- My work does not contain any instances of plagiarism and/or collusion.
  My work does not contain any fabricated data.

By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.

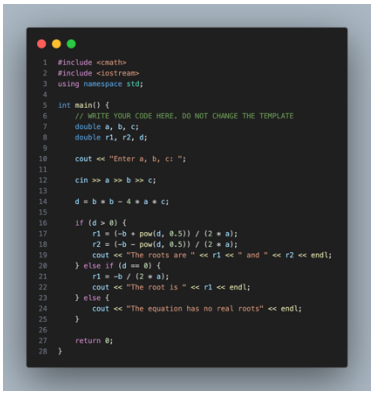
Scoring – For Tutor Use	
Student ID:2468293	**Student Name:Aibo Ge**

Stage of Marking	**Marker Code**	**Learning Outcomes Achieved （F/P/M/D） (please modify as appropriate)**			**Final Score**
		A	**B**	**C**	
1st Marker – red pen					
Moderation – green pen	**IM Initials**	The original mark has been accepted by the moderator (please circle as appropriate):			Y / N
		Data entry and score calculation have been checked by another tutor (please circle):			Y
2nd Marker if needed – green pen					

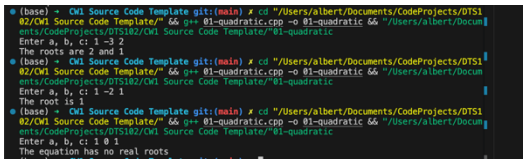
For Academic Office Use			**Possible Academic Infringement (please tick as appropriate)**	
Date Received	**Days late**	**Late Penalty**	☐ **Category A**	Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary) _____
			☐ **Category B**	
			☐ **Category C**	
			☐ **Category D**	
			☐ **Category E**	

DTS102TC Programming with C++
Coursework 1 (Assignment)

Question 1. *Algebra: solve quadratic equations* (5 marks)



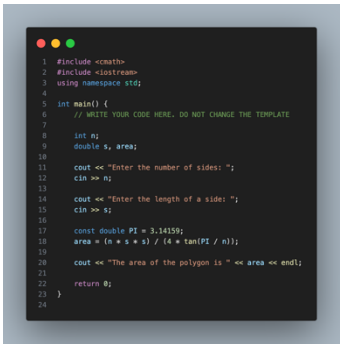
```cpp
#include <cmath>
#include <iostream>
using namespace std;

int main() {
    // WRITE YOUR CODE HERE. DO NOT CHANGE THE TEMPLATE
    double a, b, c;
    double r1, r2, d;

    cout << "Enter a, b, c: ";

    cin >> a >> b >> c;

    d = b * b - 4 * a * c;

    if (d > 0) {
        r1 = (-b + pow(d, 0.5)) / (2 * a);
        r2 = (-b - pow(d, 0.5)) / (2 * a);
        cout << "The roots are " << r1 << " and " << r2 << endl;
    } else if (d == 0) {
        r1 = -b / (2 * a);
        cout << "The root is " << r1 << endl;
    } else {
        cout << "The equation has no real roots" << endl;
    }

    return 0;
}
```



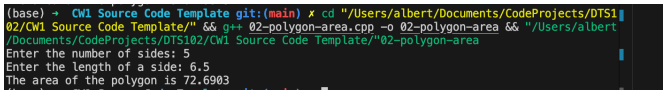
```
(base) ➜  CW1 Source Code Template git:(main) ✗ cd "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/" && g++ 01-quadratic.cpp -o 01-quadratic && "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/"01-quadratic
Enter a, b, c: 1 -3 2
The roots are 2 and 1
(base) ➜  CW1 Source Code Template git:(main) ✗ cd "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/" && g++ 01-quadratic.cpp -o 01-quadratic && "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/"01-quadratic
Enter a, b, c: 1 -2 1
The root is 1
(base) ➜  CW1 Source Code Template git:(main) ✗ cd "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/" && g++ 01-quadratic.cpp -o 01-quadratic && "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/"01-quadratic
Enter a, b, c: 1 0 1
The equation has no real roots
```

The core functionality is implemented within the main function, which solves quadratic equations of the form ax^2 + bx + c = 0. The program first reads the coefficients a, b, and c. The critical step is the calculation of the discriminant, d = b^2 - 4ac, which determines the nature of the roots.
The program uses a conditional structure (if-else if-else) to handle three scenarios:
1. If d > 0, the equation has two distinct real roots. These are calculated using the quadratic formula (-b +/- sqrt(d)) / 2a. The pow(d, 0.5) function is used to compute the square root.
2. If d = 0, the equation has exactly one real root, calculated as -b / 2a.
3. If d < 0, the equation has no real roots.
This logic ensures all possible cases for real coefficients are handled correctly.

Question 2. *Geometry: area of a regular polygon* (5 marks)



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

int main() {
    // WRITE YOUR CODE HERE. DO NOT CHANGE THE TEMPLATE

    int n;
    double s, area;

    cout << "Enter the number of sides: ";
    cin >> n;

    cout << "Enter the length of a side: ";
    cin >> s;

    const double PI = 3.14159;
    area = (n * s * s) / (4 * tan(PI / n));

    cout << "The area of the polygon is " << area << endl;

    return 0;
}
```

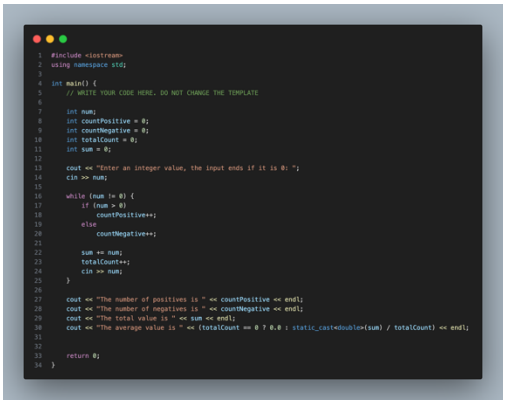


```
(base) ➜  CW1 Source Code Template git:(main) ✗ cd "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/" && g++ 02-polygon-area.cpp -o 02-polygon-area && "/Users/albert/Documents/CodeProjects/DTS102/CW1 Source Code Template/"02-polygon-area
Enter the number of sides: 5
Enter the length of a side: 6.5
The area of the polygon is 72.6903
```

*The program calculates the area of a regular polygon given the number of sides (n) and the side length (s). The mathematical formula used is Area = (n * s^2) / (4 * tan(PI / n)).*

*The implementation defines PI as a constant double for precision. It utilizes the tan function from the <cmath> library. The expression (n * s * s) / (4 * tan(PI / n)) directly translates the mathematical formula into C++ code. The result is stored in a double variable to accommodate floating-point values.*

Question 3. *Count positive and negative numbers and compute the average of numbers* (5 marks)



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

int main() {
    // WRITE YOUR CODE HERE. DO NOT CHANGE THE TEMPLATE

    int num;
    int countPositive = 0;
    int countNegative = 0;
    int totalCount = 0;
    int sum = 0;

    cout << "Enter an integer value, the input ends if it is 0: ";
    cin >> num;

    while (num != 0) {
        if (num > 0)
            countPositive++;
        else
            countNegative++;

        sum += num;
        totalCount++;
        cin >> num;
    }

    cout << "The number of positives is " << countPositive << endl;
    cout << "The number of negatives is " << countNegative << endl;
    cout << "The total value is " << sum << endl;
    cout << "The average value is " << (totalCount == 0 ? 0.0 : static_cast<double>(sum) / totalCount) << endl;


    return 0;
}
```

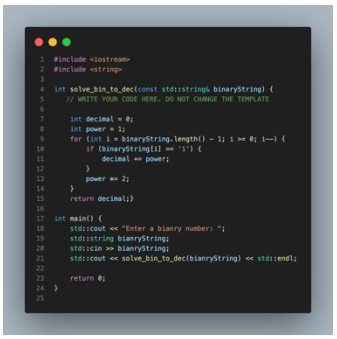
```
(base) ➜  CW1 Source Code Template git:(main) ✗ cd "/Users/albert/Documents/C
02/CW1 Source Code Template/" && g++ 03-counter.cpp -o 03-counter && "/Users/
/CodeProjects/DTS102/CW1 Source Code Template/"03-counter
Enter an integer value, the input ends if it is 0: 34 26 52 48 31 48 83 32 0
The number of positives is 8
The number of negatives is 0
The total value is 354
The average value is 44.25
```

The program analyzes a stream of integer inputs ending with 0. It employs a while loop that continues execution as long as the input is not 0 (the sentinel value).

Inside the loop, conditional statements classify each number as positive or negative, incrementing the corresponding counters (countPositive, countNegative). Every non-zero number is added to a running sum.

After the loop terminates, the average is calculated. A static_cast<double>(sum) is used to convert the integer sum to a floating-point number before division, ensuring the average is computed with decimal precision rather than performing integer division.

Question 4. *Binary to decimal* (5 marks)



```
#include <iostream>
#include <string>

int solve_bin_to_dec(const std::string& binaryString) {
    // WRITE YOUR CODE HERE. DO NOT CHANGE THE TEMPLATE

    int decimal = 0;
    int power = 1;
    for (int i = binaryString.length() - 1; i >= 0; i--) {
        if (binaryString[i] == '1') {
            decimal += power;
        }
        power *= 2;
    }
    return decimal;}

int main() {
    std::cout << "Enter a bianry number: ";
    std::string bianryString;
    std::cin >> bianryString;
    std::cout << solve_bin_to_dec(bianryString) << std::endl;

    return 0;
}
```

```
(base) ➜  CW1 Source Code Template git:(main) ✗ cd "/Users/a
02/CW1 Source Code Template/" && g++ 04-bin2dec.cpp -o 04-bi
/CodeProjects/DTS102/CW1 Source Code Template/"04-bin2dec
Enter a bianry number: 10001
17
(base) ➜  CW1 Source Code Template git:(main) ✗ cd "/Users/a
02/CW1 Source Code Template/" && g++ 04-bin2dec.cpp -o 04-bi
/CodeProjects/DTS102/CW1 Source Code Template/"04-bin2dec
Enter a bianry number: 11011111101010010000000000
58631168
(base) ➜  CW1 Source Code Template git:(main) ✗
```

The function solve_bin_to_dec converts a binary string representation into a decimal integer. The algorithm iterates through the string from the last character (least significant bit) to the first (most significant bit).
A variable 'power' tracks the place value of the current bit (1, 2, 4, 8, ...). In each iteration:
1. If the current character is '1', the current 'power' value is added to the total decimal result.
2. The 'power' is multiplied by 2 to prepare for the next bit position.
This approach efficiently converts the binary number in a single pass without the overhead of repeated calls to a power function.

Question 5. *Print distinct numbers* (5 marks)

```
#include <iostream>

int main() {
    // WRITE YOUR CODE HERE. DO NOT CHANGE THE TEMPLATE

    int numbers[10];  // Store distinct values
    int size = 0;     // Indicate how many distict values are in the array numbers

    std::cout << "Enter ten integers: ";

    int temp;
    for (int i = 0; i < 10; i++) {
        std::cin >> temp;
        bool isDistinct = true;
        for (int j = 0; j < size; j++) {
            if (numbers[j] == temp) {
                isDistinct = false;
                break;
            }
        }
        if (isDistinct) {
            numbers[size] = temp;
            size++;
        }
    }

    std::cout << "The number of distinct numbers is " << size << std::endl;
    std::cout << "The distinct numbers are:"; // Note: Sample output shows space after colon? No, sample shows "The distinct numbers are: 1 2 3 6 4 5"
    for (int i = 0; i < size; i++) {
        std::cout << " " << numbers[i];
    }
    std::cout << std::endl;

    return 0;
}
```

```
(base) ➜  CW1 Source Code Template git:(main) ✗ cd "/U
02/CW1 Source Code Template/" && g++ 05-distinct-numbe
s/albert/Documents/CodeProjects/DTS102/CW1 Source Code
Enter ten integers: 2 7 1 3 0 9 5 9 4 2 6 1 5 2
The number of distinct numbers is 8
The distinct numbers are: 2 7 1 3 0 9 5 4
```

The program reads ten integers and filters out duplicates to display only distinct numbers. It utilizes an array 'numbers' to store the unique values identified so far and a counter 'size' to track the count of these unique values.

For each new input, a nested loop performs a linear search through the 'numbers' array. A boolean flag 'isDistinct' is used to indicate if the number is already present. If the loop completes without finding the number, it is considered distinct, added to the array, and the size counter is incremented. This ensures that the output contains each number exactly once, preserving their original relative order.

Question 6. *Sum elements in each column* (10 marks)

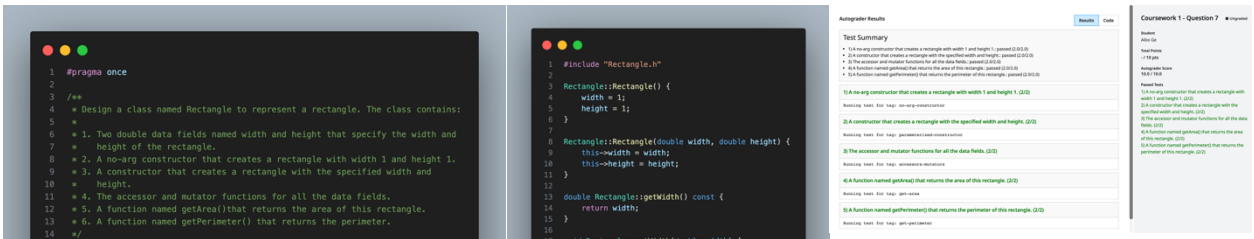
```
#include <iostream>

const int SIZE = 4;

double sumColumn(const double m[][SIZE], int rowSize, int columnIndex) {
// WRITE YOUR CODE HERE. DO NOT CHANGE THE TEMPLATE

    double sum = 0;
    for (int i = 0; i < rowSize; i++) {
        sum += m[i][columnIndex];
    }
    return sum;
}

int main() {
    double matrix[3][SIZE]{};

    std::cout << "Enter a 3-by-4 matrix row by row: " << std::endl;

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < SIZE; j++) {
            std::cin >> matrix[i][j];
        }
    }


    for (int j = 0; j < SIZE; j++)
        std::cout << "Sum of the elements at column " << j << " is " << sumColumn(matrix, 3, j) << std::endl;

    return 0;
}
```

```
(base) ➜  CW1 Source Code Template git:(m
02/CW1 Source Code Template/" && g++ 06-s
bert/Documents/CodeProjects/DTS102/CW1 So
Enter a 3-by-4 matrix row by row:
72 69 75 53
75 1 13 2
22 50 92 17
Sum of the elements at column 0 is 169
Sum of the elements at column 1 is 120
Sum of the elements at column 2 is 180
Sum of the elements at column 3 is 72
```

The function sumColumn computes the sum of elements in a specific column of a matrix. It accepts a 2D array, the number of rows, and the target column index as parameters.

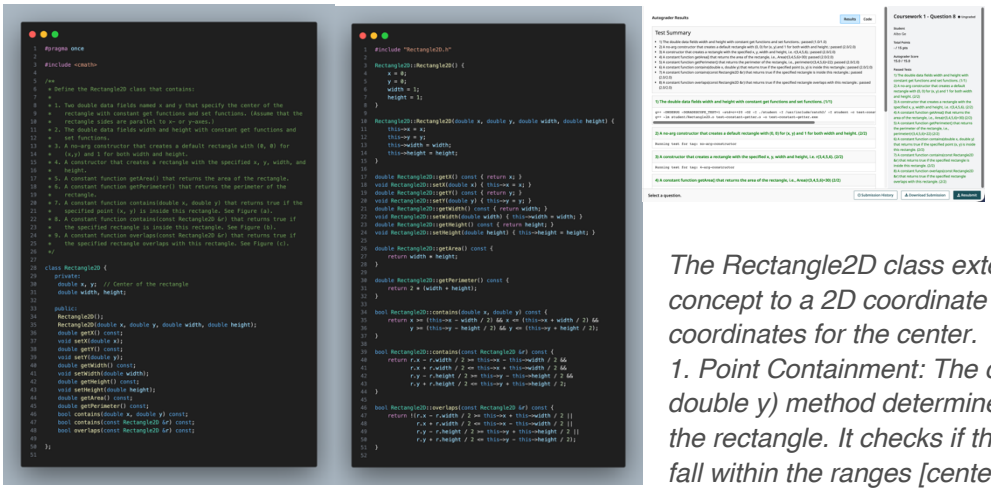
The logic involves a single for-loop that iterates through the rows (from 0 to rowSize - 1) while keeping the column index fixed. In each iteration, the value at the current row and specified column is added to an accumulator variable 'sum'. This effectively traverses the matrix vertically. The main function demonstrates this by calling sumColumn for each column index (0 to 3) of a 3x4 matrix.

Question 7. *The Rectangle class* (10 marks)



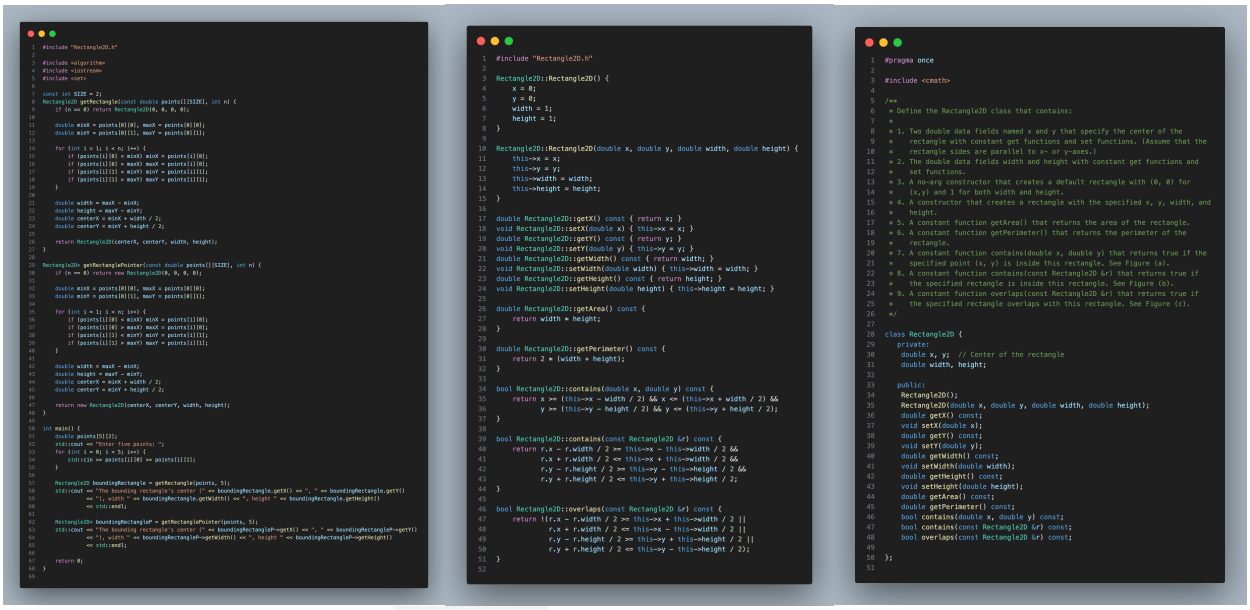
The Rectangle class encapsulates the properties and behaviors of a geometric rectangle.
1. Data Encapsulation: The width and height are stored as private double variables, ensuring they are modified only through defined interfaces.
2. Constructors: A no-argument constructor initializes a default rectangle (1x1), while a parameterized constructor allows creating a rectangle with specific dimensions.
3. Accessors: Public methods getWidth and getHeight provide read access to the private data.
*4. Computation: The getArea method returns the product of width and height, and getPerimeter returns 2 * (width + height). This design adheres to object-oriented principles by bundling data with the methods that operate on it.*

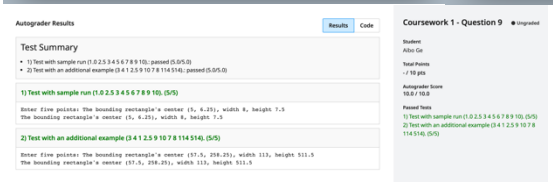
Question 8. *Geometry: The Rectangle2D class* (15 marks)



The Rectangle2D class extends the rectangle concept to a 2D coordinate space, adding (x, y) coordinates for the center.
1. Point Containment: The contains(double x, double y) method determines if a point lies inside the rectangle. It checks if the point's coordinates fall within the ranges [center_x - width/2, center_x + width/2] and [center_y - height/2, center_y + height/2].
2. Rectangle Containment: The contains(const Rectangle2D &r) method checks if another rectangle 'r' is entirely within the current rectangle. This requires that all boundaries of 'r' are within the boundaries of the current rectangle.
3. Overlap Detection: The overlaps(const Rectangle2D &r) method determines if two rectangles intersect. It uses the logic that two rectangles overlap unless they are completely separated (e.g., one is entirely to the left, right, above, or below the other).

Question 9. *Geometry: find the bounding rectangle* (10 marks)





The getRectangle function calculates the minimum bounding rectangle that encloses a given set of 2D points.
The algorithm iterates through the entire array of points to identify the extreme coordinates:
- minX: the smallest x-coordinate
- maxX: the largest x-coordinate
- minY: the smallest y-coordinate
- maxY: the largest y-coordinate
These four values define the boundaries of the bounding box. The width is calculated as (maxX - minX) and the height as (maxY - minY). The center of the bounding rectangle is then derived as the midpoint of these ranges: (minX + width / 2, minY + height / 2). This efficiently determines the smallest axis-aligned rectangle containing all points.