Swinburne University of Technology

Faculty of Science, Engineering and Technology

ASSIGNMENT COVER SHEET

Subject Code: COS30008
Subject Title: Data Structures and Patterns
Assignment number and title: 1, Solution Design in C++
Due date: Monday, October 7, 2024, 3:59
Lecturer: Dr. Ky Trung Pham

Your name: Nguyen Minh Duy **Your student ID:** 104974743

Check Tutorial	Mon 10:30	Mon 14:30	Tues 08:30	Tues 10:30	Tues 12:30	Tues 14:30	Tues 16:30	Wed 08:30	Wed 10:30	Wed 12:30	Wed 14:30

Marker's comments:

Problem	Marks	Obtained
1	38	
2	60	
3	38	
4	20	
Total	156	

Extension certification:

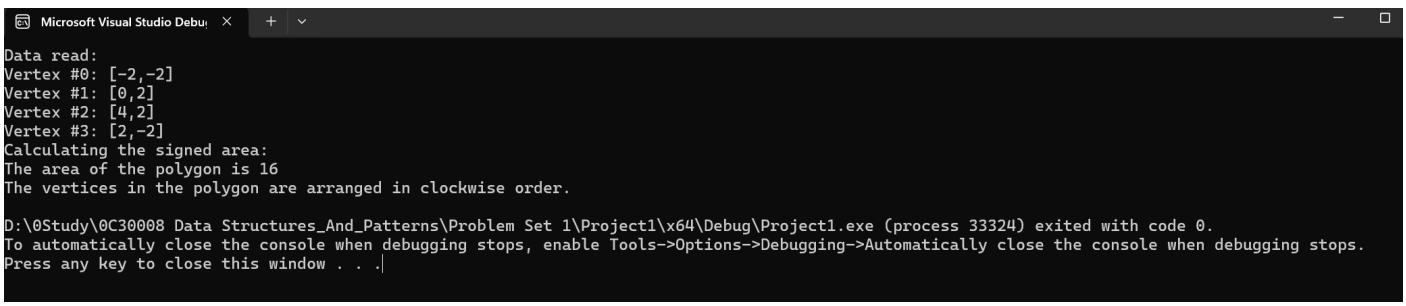
This assignment has been given an extension and is now due on _____

Signature of Convener:_____

Problem 1

...tterns\Problem Set 1\Project1\Project1\PolygonPS1.cpp 1

```cpp
#include "Polygon.h"  // Nhúng file định nghĩa cho lớp Polygon
#include <iostream>   // Thêm thư viện cho cout

using namespace std;

// Mở rộng bài tập Problem Set 1
float Polygon::getSignedArea() const
{
    float area = 0.0f; // Khởi tạo biến lưu diện tích có dấu

    // Kiểm tra xem đa giác có ít nhất ba đỉnh hay không
    if (fNumberOfVertices > 2) // Đa giác phải có ít nhất 3 đỉnh
    {
        // Duyệt qua các đỉnh của đa giác từ đỉnh thứ 1 đến đỉnh cuối
        for (size_t i = 1; i < fNumberOfVertices; i++) // Bắt đầu từ đỉnh thứ 1
        {
            Vector2D v0 = fVertices[i - 1]; // Lấy đỉnh trước đó
            Vector2D v1 = fVertices[i]; // Lấy đỉnh hiện tại

            // Tính tích chéo và cộng vào diện tích
            area += v0.getX() * v1.getY() - v0.getY() * v1.getX(); // Tích chéo
            // Có thể thay thế bằng: area += v0.cross(v1); nếu Vector2D có phương thức cross
        }

        // Tính tích chéo giữa đỉnh cuối cùng và đỉnh đầu tiên
        Vector2D v0 = fVertices[fNumberOfVertices - 1]; // Đỉnh cuối
        Vector2D v1 = fVertices[0]; // Đỉnh đầu tiên

        // Cộng diện tích của hình tứ giác được tạo ra từ đỉnh cuối và đỉnh đầu tiên
        area += v0.getX() * v1.getY() - v0.getY() * v1.getX(); // Tích chéo
        // Có thể thay thế bằng: area += v0.cross(v1); nếu Vector2D có phương thức cross
    }

    return area / 2.0f; // Chia cho 2 để có diện tích thực tế
} // Kết thúc hàm

```

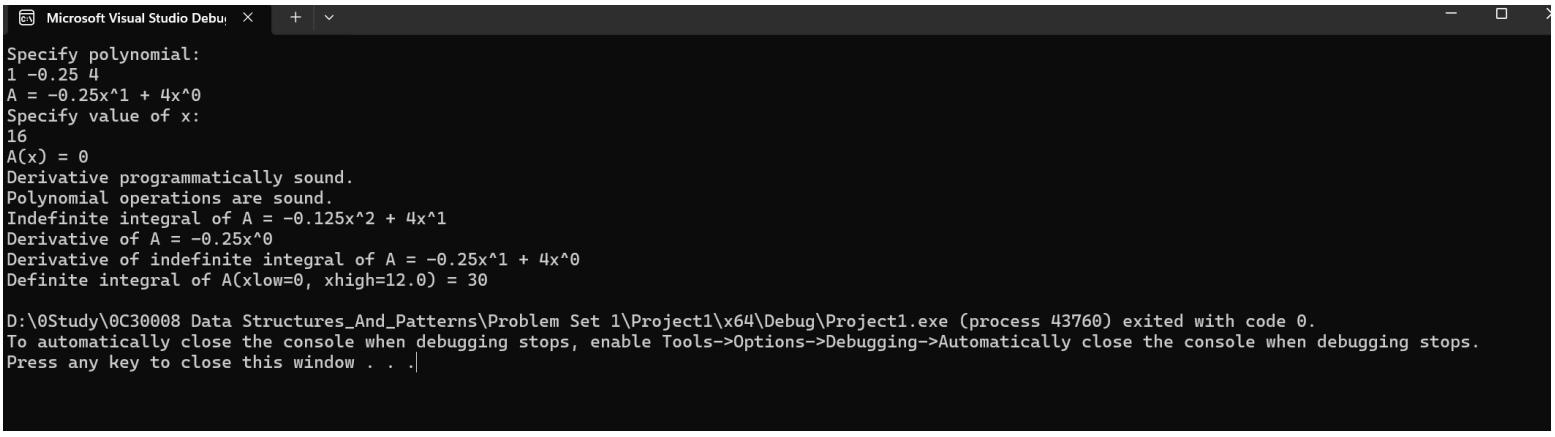


Problem 2

...rns\Problem Set 1\Project1\Project1\PolynomialPS1.cpp 1

```cpp
#include "Polynomial.h"  // Nhúng file định nghĩa cho lớp Polynomial

#include <cmath>        // Thư viện toán học để sử dụng pow()

using namespace std;

// Hàm tính giá trị của đa thức tại một giá trị x cho trước
double Polynomial::operator()(double aX) const
{
    double Result = 0.0;  // Khởi tạo biến để lưu kết quả


    // Vòng lặp qua từng hệ số của đa thức để tính giá trị
    for (size_t i = 0; i <= fDegree; i++)
    {
        Result += fCoeffs[i] * pow(aX, i);  // Tính giá trị của đa thức tại x
    }

    return Result;  // Trả về kết quả tính được

}

// Hàm tính đạo hàm của đa thức
Polynomial Polynomial::getDerivative() const
{
    Polynomial Result;  // Khởi tạo đa thức kết quả


    if (fDegree)  // Nếu bậc của đa thức lớn hơn 0

    {
        Result.fDegree = fDegree - 1;  // Cập nhật bậc cho đạo hàm


        // Vòng lặp để tính toán các hệ số của đạo hàm
        for (size_t i = 0; i < fDegree; i++)
        {
            Result.fCoeffs[i] = (i + 1) * fCoeffs[i + 1];  // Tính toán hệ số của đạo hàm
        }
    }

    return Result;  // Trả về đa thức đạo hàm

}

// Hàm tính tích phân không xác định của đa thức
Polynomial Polynomial::getIndefiniteIntegral() const
```

```
42  {
43      Polynomial Result;  // Khởi tạo đa thức cho tích phân không xác định
44
45      Result.fDegree = fDegree + 1;  // Cập nhật bậc cho đa thức tích phân
          không xác định
46
47      // Vòng lặp để tính toán các hệ số của tích phân không xác định
48      for (size_t i = 1; i <= fDegree + 1; i++)
49      {
50          Result.fCoeffs[i] = fCoeffs[i - 1] / i;  // Tính toán hệ số cho
              tích phân
51      }
52
53      return Result;  // Trả về đa thức tích phân không xác định
54  }
55
56  // Hàm tính tích phân xác định của đa thức giữa hai giới hạn
57  double Polynomial::getDefiniteIntegral(double aXLow, double aXHigh) const
58  {
59      Polynomial lIndefiniteIntegral = getIndefiniteIntegral();  // Gọi hàm
          tính tích phân không xác định
60
61      return lIndefiniteIntegral(aXHigh) - lIndefiniteIntegral(aXLow);  //
          Tính hiệu giữa hai giá trị tích phân
62  }
```



Problem 3

...terns\Problem Set 1\Project1\Project1\Combination.cpp 1

```cpp
#include "Combination.h"

// Hàm khởi tạo cho lớp Combination với các tham số aN và aK
Combination::Combination(size_t aN, size_t aK) :
    fN(aN),                                                    // Gán giá trị aN cho biến thành viên fN
    fK(aK)                                                     // Gán giá trị aK cho biến thành viên fK
{}

// Hàm trả về giá trị của fN
size_t Combination::getN() const
{
    return fN;                                                 // Trả về giá trị của biến fN
}

// Hàm trả về giá trị của fK
size_t Combination::getK() const
{
    return fK;                                                 // Trả về giá trị của biến fK
}

// Hàm toán tử để tính giá trị tổ hợp C(n, k)
unsigned long long Combination::operator()() const
{
    // Kiểm tra xem k có lớn hơn n hay không
    if (fK > fN)                                               // Nếu k lớn hơn n, không thể tính tổ hợp
    {
        return 0;                                              // Trả về 0 nếu k > n
    }
    else
    {
        unsigned long long Result = 1;                         // Khởi tạo biến Result với giá trị 1

        // Vòng lặp để tính toán tổ hợp
        for (unsigned long long i = 0; i < fK; )               // Lặp từ 0 đến k
        {
            Result *= (fN - i);                                // Nhân Result với (n - i), thực hiện tính toán tổ hợp
            Result /= ++i;                                     // Tăng i lên 1 và chia cho giá trị mới của i
        }

```

```
40          return Result;                                          // Trả
              về giá trị cuối cùng của tổ hợp C(n, k)
41      }
42  }
43
```

```
Microsoft Visual Studio Debu
The first ten levels of Pascal's triangle:
(n=0, 0<=k<=0):                          1
(n=1, 0<=k<=1):                       1     1
(n=2, 0<=k<=2):                    1     2     1
(n=3, 0<=k<=3):                 1     3     3     1
(n=4, 0<=k<=4):              1     4     6     4     1
(n=5, 0<=k<=5):           1     5    10    10     5     1
(n=6, 0<=k<=6):        1     6    15    20    15     6     1
(n=7, 0<=k<=7):     1     7    21    35    35    21     7     1
(n=8, 0<=k<=8):  1     8    28    56    70    56    28     8     1
(n=9, 0<=k<=9):  1     9    36    84   126   126    84    36     9     1

Large Numbers:
28 over 14 = 40116600
52 over 5 = 2598960

D:\0Study\0C30008 Data Structures_And_Patterns\Problem Set 1\Project1\x64\Debug\Project1.exe (process 22576) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Problem 4

... Set 1\Project1\Project1\BernsteinBasisPolynomial.cpp 1

```cpp
#include "BernsteinBasisPolynomial.h"  // Nhúng file định nghĩa cho lớp BernsteinBasisPolynomial

#include <cmath>                       // Thư viện toán học để sử dụng các hàm toán học

using namespace std;

// Hàm khởi tạo cho lớp BernsteinBasisPolynomial với tham số aV và aN
BernsteinBasisPolynomial::BernsteinBasisPolynomial(unsigned int aV, unsigned int aN) :
    fFactor(aN, aV)                                                    // Khởi tạo biến fFactor với các tham số aN và aV
{}

// Hàm toán tử để tính giá trị của đa thức Bernstein tại một giá trị x cho trước
double BernsteinBasisPolynomial::operator()(double aX) const
{
    // Tính giá trị của đa thức Bernstein với công thức C(n, k) * x^k * (1 - x)^(n - k)
    double Result = fFactor() * pow(aX, fFactor.getK());               // Tính C(n, k) * x^k

    // Nhân kết quả với (1 - x)^(n - k) và trả về giá trị cuối cùng
    return Result * pow(1 - aX, fFactor.getN() - fFactor.getK());      // Trả về giá trị của đa thức Bernstein
}

```