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: Thursday, March 24, 2022, 14:30
Lecturer: Dr. Markus Lumpe

Your name: Dao Khanh Nga Thi **Your student ID:** 104177393

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
				X							

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: PolygonPS1.cpp

```
#include "Polygon.h"

float Polygon::getSignedArea() const
{
    float result = 0.0f;

    if (fNumberOfVertices > 2) {
        for (size_t i = 0; i < fNumberOfVertices; i++) {
            size_t j = (i + 1) % fNumberOfVertices;
            float x1 = fVertices[i].getX();
            float y1 = fVertices[i].getY();
            float x2 = fVertices[j].getX();
            float y2 = fVertices[j].getY();
            result += 0.5f * (x1 * y2 - y1 * x2);
        }
    }

    return result;
}
```

Problem 2: PolynomialPS1.cpp

```
#include "Polynomial.h"
#include <cmath>

//Last 4 methods
double Polynomial::operator()(double x) const {
    if (fDegree == 0) {
        return fCoeffs[0];
    }

    double result = fCoeffs[0];
    double power = 1.0;

    for (int i = 1; i <= fDegree; i++) {
        power *= x;
        result += fCoeffs[i] * power;
    }

    return result;
}

Polynomial Polynomial::getDerivative() const {
    Polynomial result;

    if (fDegree == 0) {
        return result;
    }

    result.fDegree = fDegree - 1;

    for (int i = 1; i <= fDegree; i++) {
        result.fCoeffs[i - 1] = fCoeffs[i] * i;
    }

    return result;
}

Polynomial Polynomial::getIndefiniteIntegral() const {
    Polynomial result;
    result.fDegree = fDegree + 1;

    for (int i = fDegree + 1; i > 0; i--) {
        result.fCoeffs[i] = fCoeffs[i - 1] / i;
    }

    return result;
}

double Polynomial::getDefiniteIntegral(double xLow, double xHigh) const {
    Polynomial indefiniteIntegral = getIndefiniteIntegral();
    return indefiniteIntegral(xHigh) - indefiniteIntegral(xLow);
}
```

Problem 3: Combination.cpp

```
#include "Combination.h"

Combination::Combination(size_t aN, size_t aK) : fN(aN), fK(aK) {}

size_t Combination::getN() const {
    return fN;
}

size_t Combination::getK() const {
    return fK;
}

unsigned long long Combination::operator()() const {
    if (fK > fN) {
        return 0ull;
    }

    unsigned long long result = 1;
    size_t smaller = (fK < fN - fK) ? fK : fN - fK;
    size_t f = fN;

    for (size_t i = 1; i <= smaller; i++) {
        result *= f--;
        result /= i;
    }

    return result;
}
```

Problem 4: BernsteinBasisPolynomial.cpp

```
#include "BernsteinBasisPolynomial.h"
#include <cmath>

BernsteinBasisPolynomial::BernsteinBasisPolynomial(unsigned int aV, unsigned int aN)
:
    fFactor(Combination(aN, aV))
{}

double BernsteinBasisPolynomial::operator()(double x) const {
    double result = fFactor() * pow(x, fFactor.getK()) * pow((1 - x),
(fFactor.getN() - fFactor.getK()));
    return result;
}
```