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: Trac Duc Anh Luong **Your student ID:** 103488117

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:____________________________

```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;
            // shoelace algorithm
            Result += 0.5 * (fVertices[i].getX() * fVertices[j].getY() - 
              fVertices[i].getY() * fVertices[j].getX());
        }
    }
    return Result;
}
```

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

double Polynomial::operator()(double aX) const
{
    double result = 0.0;

    for (int i = 0; i <= fDegree; i++) {
        result += fCoeffs[i] * pow(aX, i);
    }
    return result;
}

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

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

    Result.fDegree = fDegree - 1;

    for (size_t 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; i >= 0; i--) {
        Result.fCoeffs[i + 1] = fCoeffs[i] / (i + 1);
    }

    return Result;
}

double Polynomial::getDefiniteIntegral(double aXLow, double aXHigh) const
{
    return this->getIndefiniteIntegral()(aXHigh) - this-
      >getIndefiniteIntegral()(aXLow);
}
```

```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 0ll;
    unsigned long long Result = 1;

    for (size_t i = 0; i < fK; i++) {
        Result *= (fN - i);
        Result /= (i + 1);
    }

    return Result;
}
```

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

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

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