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: Tuesday, September 24, 2024, 23:59
Lecturer: Dr. Phan Thanh Tra

Your name: Nguyen Thi Thanh Minh **Your student ID:** 104169617

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	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
// COS30008, Problem Set 1/1, 2024

#include "Polygon.h"

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

    if (fNumberOfVertices > 2) // a polygon with less than 3 vertices has no area
    {
        for (size_t i = 0; i < fNumberOfVertices; i++)
        {
            Vector2D lCurrent = fVertices[i];
            // if the current vertex is the last one, the next vertex will be the first vertice
            Vector2D lNext = fVertices[(i + 1) % fNumberOfVertices];

            // add determinant to the sum
            // in 2D, this value is mathematically equal to the cross product of the two 2D vectors in terms of magnitude (= x1 * y2 - y1 * x2)
            result += lCurrent.cross(lNext);
            // of course, we can calculate it independently as follows
            //result += lCurrent.getX() * lNext.getY() - lCurrent.getY() * lNext.getX();
        }
    }

    return result / 2.0f;
}
```

```cpp
// COS30008, Problem Set 1/2, 2024

#include "Polynomial.h"

#include <cmath>

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

    for (size_t i = 0; i <= fDegree; i++)
    {
        // raise x to the power of i, then multiply with coefficient at that degree
        result += pow(aX, i) * fCoeffs[i];
    }

    return result;
}

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

    if (fDegree > 0)
    {
        derivative.fDegree = fDegree - 1;

        for (size_t i = 0; i < fDegree; i++)
        {
            // (d/dx) a_i * x^i = i * a_i * x^(i-1)
            derivative.fCoeffs[i] = (i + 1) * fCoeffs[i + 1];
        }
    }

    return derivative;
}

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

    for (size_t i = 0; i <= fDegree; i++)
    {
        // ∫ a_i * x^k (dx) = a_i / (k+1) * x^(k+1)
        antiDer.fCoeffs[i + 1] = fCoeffs[i] / (i + 1);
    }

```

```
49      return antiDer;
50  }
51
52  double Polynomial::getDefiniteIntegral(double aXLow, double aXHigh) const
53  {
54      Polynomial antiDer = getIndefiniteIntegral();
55      return antiDer(aXHigh) - antiDer(aXLow);
56  }
```

```cpp
// COS30008, Problem Set 1/3, 2024

#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 0;
    }

    unsigned long long result = 1;

    size_t lK = fK;

    if (fK > fN / 2)
    {
        // this will reduce the number of iterations by choosing the
          smaller k
        lK = fN - fK;
    }

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

    return result;
}
```

```cpp
// COS30008, Problem Set 1/4, 2024

#include "BernsteinBasisPolynomial.h"

#include <cmath>

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

double BernsteinBasisPolynomial::operator()(double aX) const
{
    size_t lN = fFactor.getN();
    size_t lV = fFactor.getK();

    return fFactor() * pow(aX, lV) * pow(1 - aX, lN - lV);
}
```