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: Jamie Kozminska **Your student ID:** 101114436

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

// COS30008, Tutorial 2, 2022

#include "Polygon.h"

#include <stdexcept>

using namespace std;


float Polygon::getSignedArea() const
{
    double leftSum = 0.0;
    double rightSum = 0.0;

    for (int i = 0; i < fNumberOfVertices; ++i) {
        int j = (i + 1) % fNumberOfVertices;
        leftSum += fVertices[i].getX() * fVertices[j].getY();
        rightSum += fVertices[j].getX() * fVertices[i].getY();
    }

    return 0.5 * abs(leftSum - rightSum);
}

```

```cpp
// COS30008, Tutorial 3, 2022

#include "Polynomial.h"
#include <cmath>
using namespace std;


double Polynomial::operator()(double aX) const
{
    double x = 0.0;
    for (int i = 0; i <= fDegree; i++)
    {
        x += fCoeffs[i] * pow(aX, i);
    }
    return x;
}
// derivative is broken
Polynomial Polynomial::getDerivative() const
{
    //Polynomial Result;

 //   Result.fDegree = Result.fDegree - 1;

   // return Result;
    Polynomial Result;
    Result.fDegree = fDegree - 1;
    for (int i = 1; i <= fDegree; i++)
    {
        Result.fCoeffs[i - 1] = fCoeffs[i] * 1;
    }
    return Result;
}

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

    Result.fDegree = Result.fDegree + 1;
    for (int i = 0; i <= Result.fDegree; i++)
    {
        Result.fCoeffs[i+1] = fCoeffs[i] / fCoeffs[i+1];
    }


    return Result;
}

double Polynomial::getDefiniteIntegral(double aXLow, double aXHigh) const
{
```

```
50      double result = 0.0;
51      Polynomial Poly = getIndefiniteIntegral();
52      for (int i= 0; i <= Poly.fDegree; i++)
53      {
54          result += Poly.fCoeffs[i] * pow(aXHigh, i);
55          result -= Poly.fCoeffs[i] * pow(aXLow, i);
56      }
57      return result;
58  }
59
60
61      // working out, can be ignored
62      //Polynomial IndefIntegral = getIndefiniteIntegral();
63      ////this->getIndefiniteIntegral() - Result(aXHigh).getIndefiniteIntegral
          ();
64      //return 0.0f;
65  //   double widthOfRectangle = (aXHigh - aXLow) / fDegree;
66  //   double area = 0.0;
67  //   double heightOfRectangle = 0;
68  //
69  //   for (int i = 0; i < fDegree; ++i)
70  //   {
71  //       heightOfRectangle = IndefIntegral(aXLow + (i + 0.5) *
      widthOfRectangle) * IndefIntegral(aXLow + (i + 0.5) * widthOfRectangle);
72  //       area += heightOfRectangle * widthOfRectangle;  // find the area of
      the rectangle and add it to the previous area. Effectively summing up the
      area under the curve.
73  //   }
74
75
76
77
```

```cpp
#include "Combination.h"
Combination::Combination(size_t aN, size_t aK)
{
    fN = aN;
    fK = aK;
}

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

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

unsigned long long Combination::operator()() const
{
    int result = 1;
    size_t lK = fK;

    if (lK > fN - lK)
    {
        lK = fN - lK;
    }
    for (int i = 0; i < lK; ++i)
    {
        result *= (fN - i);
        result /= (i + 1);
    }

    return result;
}

```

```cpp
#include "BernsteinBasisPolynomial.h"
BernsteinBasisPolynomial::BernsteinBasisPolynomial(unsigned int aV, unsigned int aN)
{
}

double BernsteinBasisPolynomial::operator()(double aX) const
{
    double cube = (1 - aX) * (1 - aX) * (1 - aX);
    double xSquared = aX * aX;
    return (10 * xSquared) * cube;
}

```