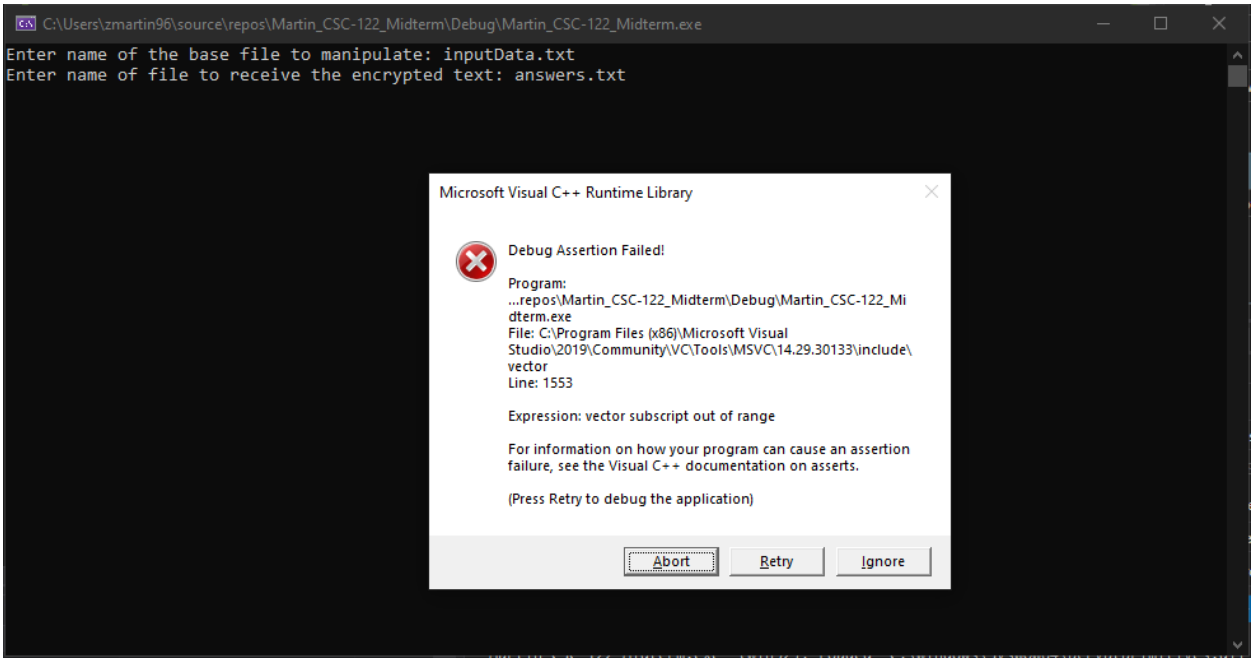

Data.h

```
#pragma once
#include "Shape.h"
#include <string>
using namespace std;

class Data
{
private:
	ifstream inFile;
	ofstream outFile;
public:
	Data() {}
	Data(const string& inFileName, const string& outFileName);
	Shape* readData(int i);
};
```

Data.cpp:

```
#include "Data.h"
#include <vector> //using an array did not fix the problem

Data::Data(const string& inFileName, const string& outFileName)
{
    int i=0;
    inFile.open(inFileName);
    outFile.open(outFileName);

    if (!inFile) throw string("The file in could not be opened!\n\n");
    if (!outFile) throw string("The file out could not be opened!\n\n");

    vector<Shape*> shapes; //new vector of shapes
    //said "vector subscript is out of range" but still threw an
    //exception when I switched to arrays
    //using smart pointers had no noticable effect

    while (!inFile.fail())
    {
        //This is where I was getting an error when running
        shapes[i]=this->readData(i);
        /*Unhandled exception at 0x7C331856 (ucrtbased.dll) in Martin_CSC-
122_Midterm.exe:
        An invalid parameter was passed to a function that considers invalid parameters
fatal.*/
        i++;
    }
    for (int j = 0; j < i + 1; j++)
    {
        outFile << "Area: " << shapes[j]->getArea()
            << "\tPerimeter: " << shapes[j]->getPerimeter()
            << endl;
    }
    //this->outFile << "Area: " << r->getArea() << "\tPerimeter: " << r->getPerimeter();

    inFile.close();
    outFile.close();
}

Shape* Data::readData(int i)
{
    if (i % 2 == 0||i==0) //is even
    {
        Rectangle* r = new Rectangle;
        inFile >> *r; //switching overloaded insertion operator to use pointers yielded
the same error
        return r;
    }
    else // is odd
    {
        Circle* c = new Circle;
        inFile >> *c;
        return c;
    }
    throw string("There was an error reading the data\n"); //didn't throw
}
```

Shapes.h

```
#include <string>
#include <iostream>
#include <fstream>
#include "Data.h"
#include <cmath>
using namespace std;

class Shape
{
protected:
    double perimeter, area;
public:
    //Shape(const string& inFileName, const string& outFileName);
    static const double PI;
    double getArea() { return this->area; }
    double getPerimeter() { return this->perimeter; }
    virtual void calcPerimeter() = 0;
    virtual void calcArea() = 0;

    //Overloaded stream insertion
    //
};

//Rectangle class
class Rectangle : public Shape
{
private:
    double length, width;
    virtual void calcPerimeter() override
    {
        this->perimeter = 2 * (this->length + this->width);
    }
    virtual void calcArea() override
    {
        this->area = this->length * this->width;
    }
public:
    friend istream& operator>>(istream& in, Rectangle&);
        //Rectangle(const string& inFileName, const string& outFileName,double l,double
w)
        //: Shape(inFileName, outFileName)
        //{
        //    this->length=l;
        //    this->width = w;
        //}

};

//Circle class
class Circle : public Shape
{
private:
    double radius=0;
    virtual void calcPerimeter() override
```

```
    {
        this->perimeter = 2 * this->radius * PI;
    }
    virtual void calcArea() override
    {
        this->area = pow(this->radius, 2) * PI;
    }
public:
    friend istream& operator>>(istream& in, Circle&);
    //Circle(const string& inFileName, const string& outFileName, double r)
    //    : Shape(inFileName, outFileName)
    //{
    //    this->radius=r;
    //}
};
```

EndUser.cpp

```cpp
#include "Data.h"

//pi
const double Shape::PI = 3.141592653359;

//overloaded stream insertion operator for Rectangle
istream& operator>>(istream& in, Rectangle& a)
{
    in >> a.length;
    in >> a.width;
    if (a.length == 50 || a.width == 50) throw string("Not Permitted Data Detected.");
    return in;
}

//overloaded stream insertion operator for Circle
istream& operator>>(istream& in, Circle& a)
{
    in >> a.radius;
    if (a.radius == 50) throw string("Not Permitted Data Detected.");
    return in;
}

int main()
{
    try
    {
        string inFileName,
            outFileName;
        cout << "Enter name of the base file to manipulate: ";
        cin >> inFileName;
        cout << "Enter name of file to receive the encrypted text: ";
        cin >> outFileName;
        Data d(inFileName, outFileName); //files entered fine and try-catch block worked
    }

    catch (string s)
    {
        cout<<s;
        exit(-1);
    }

    return 0;
}
```