LASALLE COLLEGE VANCOUVER

# Templates & Introduction to STL: Vectors

VGP 131 - Object Oriented Programming in C++ II

Instructor: Ivaldo Tributino

April 8, 2022

## ASSIGNMENT INSTRUCTION

- The assignment must be submitted by April 17, 2022.
- Each problem presents its own score, the sum of all scores is 100.

Student's Number:
Student's Name:

## (10 POINTS) PROBLEM 1

Consider the definition of the following function template:

```cpp
template<class Type>
Type funcExp(Type list[], int size) {
    int j;
    Type x = list[0];
    Type y = list[size - 1];
    for (j = 1; j < (size - 1)/2; j++)
    {
        if (x < list[j])
            x = list[j];
        if (y > list[size - 1 - j])
            y = list[size - 1 - j];

    }
    return x + y;
}
```

Further suppose that you have the following declarations:

```
int list[10] = {5, 3, 2, 10, 4, 19, 45, 13, 61, 11};
std::string strList[ ] = {"One", "Hello", "Four", "Three", "How", "Six"};
```

What is the output of the following statements?

```
std::cout << funExp(list, 10) <<std::endl;
std::cout << funExp(strList, 6) << std::endl;
```

## (10 Points) Problem 2

Considering operator+ overloading as shown below.

```
Polygon Polygon :: operator+(const Polygon & obj){

  Polygon temp;

  temp.length_ = length_ + obj.length_;
  temp.numberSides_ = numberSides_ + obj.numberSides_;

  return temp;
}
```

And the fallowing function template:

```
template<class Type>
Type sum(Type& x , Type& y){
  return x+y;
}
```

Further suppose that you have the following declarations:

```
Polygon p1(1,3);
Polygon p2(2,3);
Polygon p3 = sum(p1,p2);
```

What is the output of the following statements?

```
std::cout << "area:" << p3.area() <<std::endl;
```

## (10 Points) Problem 3

The operator+ can also be overloaded as a nonmember function of the class. In this situation, the operator function operator+ has two parameters. Overload the operator as a nonmember function for the class Polygon.

```
friend Polygon operator+(const Polygon &obj1, const Polygon &obj2);
```

## (10 Points) Problem 4

Overload the insertion operator, <<, and the extraction operator, >> for the Polygon class to get the following output.

```
Enter a Polygon in the form (length, number of sides)
Default Constructor Invoked
9
24
Length = 9cm, Number of sides = 24
Polygon destroyed
```

From:

```
cout << "Enter␣a␣Polygon␣in␣the␣form␣(length,␣number␣of␣sides)␣" << '\n';

Polygon poly;

std::cin >> poly;  //enter two numbers

cout << poly << endl;
```

## (10 Points) Problem 5

Write a C++ Program to Swap data using function template.

## (10 Points) Problem 6

Build a calculator class template and test it with the largeIntegers class from the previous week.

## (10 Points) Problem 7

Create a function called fibonacciSequence where the input is a n integer greater than 0($n > 0$) and the output is an vector<int> holding the first n elements of the Fibonacci sequence.

## (10 Points) Problem 8

Suppose that you have the following declarations:

```
int primeArray [] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};

// The statement:
vector<int> vecPrime(10);

// ostream_iterator
ostream_iterator<int> screen(cout, "-");

copy(primeArray, primeArray + 10, vecPrime.begin());
copy(vecPrime.rbegin() + 2, vecPrime.rend(), vecPrime.rbegin());
copy(vecPrime.rbegin() + 2, vecPrime.rend(), screen);
```

What is the output?

## (10 Points) Problem 9

Create a program that can inform the occurrence of the names Romeo and Juliet in the file RomeoAndJuliet.txt. (Hint: create a vector with all the words from the RomeoAndJuliet.txt file and then use the std ::count function).

## (10 Points) Problem 10

Let's create a vector "v" of type std::vector<std::vector<Polygon*>* > as below.

```
Polygon* p1 = new Polygon(1,3);
Polygon* p2 = new Polygon(2,3);
Polygon* p3 = new Polygon(3,3);
Polygon* p4 = new Polygon(1,4);
Polygon* p5 = new Polygon(2,4);
Polygon* p6 = new Polygon(3,4);
Polygon* p7 = new Polygon(1,5);
Polygon* p8 = new Polygon(2,5);
Polygon* p9 = new Polygon(3,5);

std::vector<Polygon*> v1{p1,p2,p3};
std::vector<Polygon*> v2{p4,p5,p6};
std::vector<Polygon*> v3{p7,p8,p9};

std::vector<std::vector<Polygon*>* > v;

v.push_back(&v1);
v.push_back(&v2);
v.push_back(&v3);
```

Now create a program using vector "v" to get the following output:

```
triangle Area: 0.433013 | Polygon destroyed
triangle Area: 1.73205 | Polygon destroyed
triangle Area: 3.89711 | Polygon destroyed

square Area: 1 | Polygon destroyed
square Area: 4 | Polygon destroyed
square Area: 9 | Polygon destroyed

pentagon Area: 1.72048 | Polygon destroyed
pentagon Area: 6.88191 | Polygon destroyed
pentagon Area: 15.4843 | Polygon destroyed
```