LASALLE COLLEGE VANCOUVER

Multithreading

VGP 131 - Object Oriented Programming in C++ II

Instructor: Ivaldo Tributino

June 6, 2022

ASSIGNMENT INSTRUCTION

- The exam must be submitted by June 12, 2022.
- Each problem presents its own score, the sum of all scores is 100.

Student's Number:
Student's Name:

(10 POINTS) PROBLEM 1

What is the different between join( ) and detach( ) for multi threading in C++?

(15 POINTS) PROBLEM 2

Can you explain why the program below generates an error or the output is always 16 instead of a pseudo-random double? Also, correct the code in main() so that the output is a pseudo-random double(Do not modify the function randomPolygon).

```
void randomPolygon(Polygon& poly){
    poly.setNumberSides(rand() % 100 + 1);
}

int main(){

    Polygon poly(4);
    double length = 4.0;
    std::thread t(randomPolygon, poly);
    if(t.joinable()) t.join();
    std::cout << poly(length) << std::endl;
```

```
   return 0;
}
```

(15 POINTS) PROBLEM 3

Complete the code in main() to get the following output:

Output:
Default Constructor Invoked
pentagon
Polygon destroyed

```
void unique_ptr_poly(std::unique_ptr<Polygon> p){
    (*p).setNumberSides(5);
    cout << p->shapeName() <<endl;
};

int main(){

    std::unique_ptr<Polygon> smart_ptr(new Polygon);
    std::thread t(unique_ptr_poly, smart_ptr);

   return 0;
}
```

(20 POINTS) PROBLEM 4

Use mutex and condition_variable to make getValue function wait alert from setPair function.

```
std::mutex m;
std::condition_variable cv;
bool dataReady{false};

map<int,string> gameMap = {{1,"Forza_Horizon_5"},{2,"Solar_Ash"}, {3, "
    Age_of_Empires_4"}};

void getValue(int key){
    cout << "I␣must␣wait␣for␣setPair" << endl;


    cout << gameMap.at(key) << endl;

}
void setPair(int x, string s){
```

```
    gameMap.insert(pair<int,string>(x,s));
    cout << "setpair: Data is ready."  << endl;

}

int main(){

    std::thread gamet1(getValue,4);
    std::thread gamet2(setPair,4,"Deathloop");

    if(gamet1.joinable()) gamet1.join();
    if(gamet2.joinable()) gamet2.join();

   return 0;
}
```

Output:
I must wait for getPair
setpair: Data is ready.
Deathloop

(20 POINTS) PROBLEM 5

Rewrite the program below, but using asynchronous tasks(std::async).

```
unsigned long long acm1 = 0;
unsigned long long acm2 = 0;
std::thread t1([&acm1, &v] {
    for (unsigned int i = 0; i < v.size() / 2; ++i)
    {
        acm1 += v[i];
    }
});
std::thread t2([&acm2, &v] {
    for (unsigned int i = v.size() / 2; i < v.size(); ++i)
    {
        acm2 += v[i];
    }
});
t1.join();
t2.join();
std::cout << "acm1: " << acm1 << endl;
std::cout << "acm2: " << acm2 << endl;
std::cout << "acm1 + acm2: " << acm1 + acm2 << endl;
```

(20 POINTS) PROBLEM 6

Implement a thread-safe queue.

```
template<typename T>
class threadsafe_queue
{
private:
    mutable std::mutex mut;
    std::queue<std::shared_ptr<T> > data_queue;
    std::condition_variable data_cond;
public:

}
```