UFPB - CI - Terceira Prova - POO - 2024.2 - Prof. Carlos Eduardo Batista

1. (3,0) Analise o código abaixo:

```
class Complexo {
private:
    double real, imag;
public:
     Complexo(double r = 0, double i = 0) :
real(r), imag(i) {}

        Complexo operator+(const Complexo&
outro) const {
         return Complexo(real + outro.real,
imag + outro.imag);
    }
        // Implemente o operador * para
multiplicação de números complexos

       friend std::ostream& operator<<
(std::ostream& os, const Complexo& c) {
        os << c.real;
        if(c.imag >= 0) os << "+" << c.imag
<< "i";
        else os << c.imag << "i";
        return os;
    }
};

int main() {
   Complexo c1(3, 4);
   Complexo c2(1, 2);
   std::cout << "c1 = " << c1 << std::endl;
   std::cout << "c2 = " << c2 << std::endl;
   Complexo soma = c1 + c2;
    std::cout << "Soma = " << soma <<
std::endl;
   Complexo produto = c1 * c2;
    std::cout << "Produto = " << produto <<
std::endl;

    return 0;
}
```

a) Implemente o operador de multiplicação (`*`) para a classe `Complexo`. Lembre-se que a multiplicação de números complexos segue a fórmula:
`(a + bi) * (c + di) = (ac - bd) + (ad + bc)i`

b) Explique por que o operador `<<` é implementado como uma função `friend` enquanto o operador `+` é implementado como um método da classe. Discuta as implicações de cada abordagem na sobrecarga de operadores.

2. (3,0) Os templates são um mecanismo fundamental na programação genérica em C++.
a) Modifique a classe `Complexo` da questão 1 para utilizar templates, permitindo que ela trabalhe com qualquer tipo numérico (int, float, double). Mostre como você declararia e implementaria a classe com templates.
b) Explique a diferença entre templates de função e templates de classe. Forneça um exemplo de cada um, destacando quando seria mais apropriado usar um ou outro.

3. 1. (3,0) Analise o código abaixo:

```
int dividir(int a, int b) {
      if      (b      ==      0)      throw
std::runtime_error("Divisão por zero");
    return a / b;
}
void processar(int x, int y) {
    try {
         std::cout << "Res: " << dividir(x,
y) << std::endl;
    }
    catch(const std::exception& e) {
          std::cout << "Erro: " << e.what()
<< std::endl;
         throw;
    }
}

int main() {
    try {
        processar(10, 2);
        processar(8, 0);
        processar(6, 3);
    }
    catch(const std::runtime_error& e) {
         std::cout << "Exceção capturada: "
<< e.what() << std::endl;
    }
       std::cout << "Fim do programa" <<
std::endl;
    return 0;
}
```

a) Qual será a saída exata deste programa? Explique o fluxo de execução e como as exceções são tratadas.

b) O que aconteceria se a linha `throw;` fosse removida do bloco catch na função `processar()`? Explique as implicações dessa modificação.