Gabarito da primeira prova de INF213

Questão 1

-- LETRA A --

```
class Func {
public:
    Func(string c, string n , double s);
    string getCPF() const;
    void setCPF(string c);
    string getNome() const;
    void setNome(string n);
    double getSalario() const;
    void setsalario(double s);

private:
    string CPF;
    string nome;
    double salario;
};


class Func20Horas : public Func {
  public :
    Func20Horas(string c, string n, double s);
    double CalcSalario();

  private:
};

class Func40HorasDE : public Func {
  public :
    Func40HorasDE(string c, string n, double s, double de);
    double getDE() const;
    void setDE(double de);
    double CalcSalario();

  private:
    double AdicionalDE;
};
```

-- LETRA B --

```
double Func20Horas::CalcSalario(){
   return getSalario();
}

double Func40HorasDE::CalcSalario(){
   return (getSalario() + AdicionalDE);
}
```

-- LETRA C --

Para obter o comportamento polimórfico da função CalcSalario, inclua na classe Func a função virtual CalcSalario() da seguinte forma:

```
virtual double CalcSalario() = 0; // função virtual pura

ou

virtual double CalcSalario() { }
```

```
void ImpSalarios(Func *v[], int n) {
    for (int i = 0; i < n; i++) {
        cout << "CPF     = " << v[i]->getCPF() << endl;
        cout << "Salario = " << v[i]->CalcSalario() << endl;
        cout << endl;
    }
}
```

-- LETRA D --

```
void ImpMaraja(Func *v[], int n) {
    int m=0;  // posicao do funcionario com maior salario
    double ms = v[m]->getSalario();
    for (int i = 1; i < n; i++)
        if (v[i]->CalcSalario() > v[m]->CalcSalario()) m = i;

    cout << "O maior salario eh de um funcionario ";
    Func20Horas *f20 = dynamic_cast<Func20Horas *>(v[m]);
    if (f20)
        cout << "20 horas \n";
    else {
        Func40HorasDE *f40 = dynamic_cast <Func40HorasDE *>(v[m]);
        cout << "40 horas com DE \n";
    }

    cout << "CPF     = " << v[m]->getCPF() << endl;
    cout << "Salario = " << v[m]->CalcSalario() << endl;
    cout << endl;
}
```

Questão 2

-- LETRA A --

ERRO: Não compila – falta o construtor default

Correção: na classe Q2 faça

Q2(int n = **0**) { ptr = new T[n]; dim=n;}

Ou

Q2( ) { }

Após a correção será impresso

```
A
```

-- LETRA B --

ERRO: O programa aborta – como não há a sobrecarga do operador de atribuição, após a atribuição x = b, tanto x->ptr quanto b->ptr irão apontar para uma mesma área e ao fim do escopo do objeto x (ao final da chave), o destrutor de x será executado e a área para onde x->ptr aponta será desalocada. Daí, ao executar b.get(0), esta função tentará retornar o valor de b->ptr

Correção: inclua a sobrecarga do operador de atribuição na classe Q2

Inclua a declaração na classe Q2

```
Q2<T> &operator=(const Q2<T> &);  // sobrecarga do operador=
```

E a implementação

```
Q2<T>& Q2<T>::operator=(const Q2<T> &q) {
    if (this != q) {
        delete [] ptr;
        dim = q.dim;
        ptr = new T[dim];
        for (int i=0; i < dim; i++)
            ptr[i] = q.ptr[i];
    }
}
```

Após a correção será impresso

```
B
Desalocou ptr
B
```

-- LETRA C --

ERRO: O programa aborta – como há o construtor de cópia da classe Q2, após a a chamada da função f(c) em que a passagem do parâmetro é por valor, tanto u->ptr (na função f) quanto c->ptr irão apontar para uma mesma área e ao fim do escopo do objeto u (ao final da função), o destrutor de u será executado e a área para onde u->ptr aponta será desalocada. Daí, ao executar c.get(0), esta função tentará retornar o valor de c->ptr

Correção: inclua a sobrecarga do operador de atribuição na classe Q2

Inclua a declaração na classe Q2

```
// CORRECAO LETRA C
Q2(const Q2<T> &);  // construtor de copia
```

E a implementação

```
template <class T>
Q2<T>::Q2(const Q2<T> &q) {
    dim = q.dim;
    ptr = new T[dim];
    for (int i=0; i < dim; i++)
        ptr[i] = q.ptr[i];
}
```

Após a correção será impresso

```
2
Desalocou ptr
Desalocou ptr
Desalocou ptr
Desalocou ptr
```

Questão 2

-- LETRA A --

ERRO: Não compila – o objeto da classe base m **NÃO PODE** ser atribuído ao objeto da classe devivada n

-- LETRA B --

RESULTADO

```
p = 5  10 -- p.calc = 50
q = 3  2  2  % -- q.calc = 2.5
r = 3  2 -- r.calc = 2.5
p = 3  2 -- p.calc = 6
q = 3  2  2  % -- q.calc = 2.5
r = 3  2 -- r.calc = 6
```

-- LETRA C --

RESULTADO

```
a = 10  20 -- a.calc = 200
b = 4  3  2  # -- b.calc = 3.5
a = 4  3 -- a.calc = 3.5
b = 4  3  2  # -- b.calc = 3.5
```

-- LETRA A --

ERRO: Não compila – um apontador para um objeto da classe base m2 **NÃO PODE** ser atribuído a um apontador para um objeto da classe devivada m3