UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
(DAELN)

Curso: Engenharia Eletrônica, **Disciplina:** Fundamentos de Programação II (IF62C), **Turma:** S__
Professores: Hermes Del Monego ( ), Jean Simão ( ), Robinson Vida ( ) **Data:** _________.

Aluno: **Código:**_____, **Início** __:__ **Fim** __:__

2ª Prova

1) $[2{,}5\,pt]$ No quadro a seguir, uma implementação em C++ de uma classe Pessoa. Informações tais como data de nascimento, idade e nome de uma pessoa estão representadas por meio dos atributos dessa classe. Esta representação pode ser melhorada.

Pessoa.h

```
#include <stdio.h>
class Pessoa {
private:
   int diaP;  int mesP;  int anoP;  int idadeP;
   char nomeP [30];
public:
   Pessoa ( int diaNa, int mesNa, int anoNa, char
nome[] );
   void Calc_Idade ( int diaAT, int mesAT, int
anoAT );
   int informaIdade ( );
};
```

Pessoa.cpp

```
#include "Pessoa.h“
#include <string.h>
Pessoa::Pessoa(int diaNa, int mesNa, int
anoNa, char nome[])
{
   idadeP = 0; diaP = diaNa; mesP = mesNa;
anoP = anoNa;
   strcpy(nomeP, nome);
}
void Pessoa::Calc_Idade (int diaAT, int mesAT,
int anoAT)
{
    idadeP = anoAT - anoP;
    if (mesP < mesAT)
    {
       idadeP = idadeP - 1;
    }
    else
    {
       if (mesP == mesAT)
       {
          if (diaP < diaAT)
          {
            idadeP = idadeP - 1;
          }
       }
    }
       printf("A idade da Pessoa %s seria %d \n",
nomeP, idadeP);
}
```

A seguir, com a finalidade de melhorar essa representação, uma lista de itens compõe essa questão:

a) O conceito de Coesão e desacoplamento foi pouco aplicado na definição dessa classe Pessoa. Indicar (circular no código) ou descrever a causa desse problema de coesão e desacoplamento na implementação da classe Pessoa.

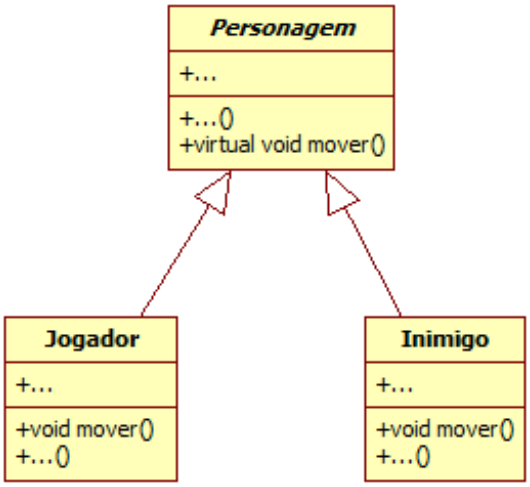
b) Modificar a classe de tal forma a respeitar o uso do conceito de Coesão e desacoplamento. Para tal, definir novos métodos.

c) Definir uma outra função *Calc_Idade* que receba apenas um parâmetro correspondente à informação do ano atual (eg. void *Calc_Idade(int anoAT)*). Fazer com que essas duas funções *Calc_Idade* possam reaproveitar ou compartilhar códigos.

d) Definir um destrutor para essa classe, bem como constutor sem parâmetro ou equivalente.

e) Existem outras formas de representar essa solução. Uma dessas outras formas seria por meio de duas classes a saber: i) classe Pessoa que armazena apenas o nome da pessoa e ii) classe Data que armazena as informações de uma data qualquer. Apresentar o Diagrama de Classes dessas duas classes de tal forma a não haver perda de funcionalidade.

2) $[2{,}5\,pt]$ Dado o diagrama de classes proposto, defina código instanciando objetos e com lógica tal que permita um quadro de polimorfismo.



Para definir esse código, um conjunto de instruções ou comandos são apresentados no quadro abaixo. O resultado desejado da execução desse código também é fornecido nesse quadro. Enfatiza-se aqui que algumas dessas instruções listadas irão apresentar erros se inseridas em um programa. Você não pode selecionar essas instruções que causam erros de compilação.

A solução desta questão consiste em ordenar de 1 a N as instruções que deverão ser colocadas dentro da função principal (main). As instruções que não receberem números não são selecionadas para compor a solução. A ordem definida na solução deverá possibilitar o resultado da execução do programa apresentado nesta questão. O número 1 ao lado da instrução significará que ela deverá ser colocada por primeiro, o número 2 indica que a instrução deverá ser inserida por segundo no programa main e assim sucessivamente.

O resultado da execução	Instruções a serem selecionadas e ordenadas
mover de p:mover do personagem. mover de p2:mover do Inimigo. mover de p3:mover do Jogador. mover de p4:mover do personagem. mover de i:mover do Inimigo. mover de k:mover do Jogador. Fim.	( ) Personagem p; Personagem *p2, *p3, p4; Inimigo *i, j; Jogador k; ( ) p4 = new Personagem(); ( ) &p4 = new Personagem(); ( ) p2 = *p4; ( ) p2 = p4; ( ) p2 = new Inimigo(); ( ) i = p2; ( ) i = new Inimigo(); ( ) p4 = &i; ( ) p4 = i; ( ) p3 = &k; ( ) std::cout << "\n mover de p:" ; p.mover(); std::cout << "\n mover de p2:" ; p2->mover(); std::cout << "\n mover de p3:" ; p3->mover(); std::cout << "\n mover de p4:" ; p4.mover(); std::cout << "\n mover de i:" ; i->mover(); std::cout << "\n mover de k:" ; k.mover(); std::cout << "\n Fim. \n";

3) $_{[2{,}5\,pt]}$ Deseja-se modelar um pequeno sistema de representação acadêmica. Este sistema deverá representar informações das matérias e dos estudantes matriculados. Tal sistema deve ser capaz de armazenar as matérias que cada aluno se matriculou e quais estudantes se matricularam na matéria. Estas informações formam o histórico escolar. Baseado no texto, pede-se: Modelar um diagrama de classes completo para o projeto deste sistema.

Informações importantes		
Estudante nome:String registro academico:int semestr de ingresso:int ano de ingresso:lon int	Materia nome:String carga horraria semanal:int	Histórico Escolar aluno:String disciplina:String semestre em que cursou ano em que cursou nota:double

4) $_{[2{,}5\,pt]}$ Utilizando os princípios de Templates em $C++$, dada a figura abaixo, transformar a classe calculadora de números inteiros em uma classe template capaz de instanciar um objeto parametrizado com qualquer tipo numérico (e.g. int, double, float). Modificar também o *main()* de maneira que se possa instanciar tais objetos de maneira correta.
Obs: Ignorar potenciais erros de instânciação caso o parâmetro fornecido não seja numérico ou não tenha os operadores pertinentes sobrecarregados.

Classe calculadora	Main

#include<iostream>
class calc
{
public:
int x;
int y;
public:
calc();
calc(int X, int Y);
~calc();
int multiplica();
int soma();
int subtrai();
int divide();
void setX(int X);
void setY(int Y);
int getX();
int getY();
};
calc::calc() {
x=0;
y=0; }
calc::calc(int X, int Y) {
x=X;
y=Y; }
calc::~calc() {
std::cout << "destruindo....."; }
int calc::multiplica(){
return x*y;}
int calc::subtrai(){
return x-y;}
int calc::divide(){
return x/y;}
int calc::soma(){
return x+y;}
void calc::setX(int X){
x=X;}
void calc::setY(int Y){
y=Y;}
int calc::getX(){
return x;}
int calc::getY(){
return y;}
	int main()
	{
	calc calcint1(2,2);
	
	calc calcint2;
	calcint2.setX(2);
	calcint2.setY(2);
	
	std::cout << calcint.soma()<<"\n";
	std::cout << calcint.subtrai()<<"\n";
	std::cout << calcint.divide()<<"\n";
	std::cout << calcint.multiplica()<<"\n";
	
	std::cout << calcint2.soma()<<"\n";
	std::cout << calcint2.subtrai()<<"\n";
	std::cout << calcint2.divide()<<"\n";
	std::cout << calcint2.multiplica()<<"\n";
	}

5) $[1{,}0\,pt]$ Uma classe abstrata pode ser diretamente instanciada? Justifique sua resposta.