Segunda Prova de Linguagens de Programação
- DCC024 -
Sistemas de Informação

Nome: _____
"Eu dou minha palavra de honra que não trapacearei neste exame."

Número de matrícula: _____

As regras do jogo:

- A prova é sem consulta.
- Quando terminar, não entregue nada além do caderno de provas para o instrutor.
- Quando escrever código, a sintaxe correta é importante.
- Cada estudante tem direito a fazer uma pergunta ao instrutor durante a prova. Isso inclui "Quanto tempo falta para acabar a prova". Para a pergunta: "Posso ir ao banheiro", a resposta é sim (deixe o telefone celular sobre a mesa para ir ao banheiro).
- A prova termina uma hora e quarenta minutos após seu início.
- Seja honesto e lembre-se: **você deu sua palavra de honra**.

Alguns conselhos:

- Escreva sempre algo nas questões, a fim de ganhar algum crédito parcial.
- Se não entender a questão, e já tiver gasto sua pergunta, escreva a sua interpretação da questão junto à resposta.
- A prova não é difícil, ela é divertida, então aproveite!

Tabela 1: Pontos acumulados (para uso do instrutor)

Questão 1	Questão 2	Questão 3	Extra

Questão Extra (0.5 Pontos): O nome do diretor de "Cidade de Deus".

1. Esta questão é sobre a diferença dos mecanismos que Python e C++ oferecem para a programação orientada a objetos.

   (a) (2 Pontos) Considere o programa abaixo, escrito em C++. Use o espaço à direita para traduzir este programa para Python.

```
class Ponto {
  private:
    int x;
  public:
    Ponto(int v) : x(v) {
    }
    virtual int
    dobro() const {
        return 2 * x;
    }
};
```

   (b) (2 Pontos) Haveria algo equivalente aos modificadores de acesso `public` e `private` em Python? Justifique a sua resposta com um exemplo, ou com uma explicação sucinta.

   (c) (2 Pontos) O que significa a palavra `virtual` em C++?

   (d) (2 Pontos) O alvo de chamadas virtuais em C++ pode ser resolvido em $O(1)$. Por que isso não é possível em Python?

   (e) (2 Pontos) Essa ideia de chamadas virtuais surgiu há bastante tempo atrás, nos idos dos anos 60. Qual linguagem a introduziu?

2. A próxima questão se refere a história de algumas linguagens de programação.

   (a) (5 Pontos) Fortran surgiu como a principal linguagem de sistemas nos anos 1950, mas acabou perdendo espaço, primeiro para C, nos anos 1970, e depois para C++, nos anos 1980. Desde então, C++ se tornou a linguagem dominante em sistemas como kernels, servidores, jogos eletrônicos, navegadores, entre outros. Mesmo linguagens muito populares, como Java, não conseguiram deslocar esse domínio. Por que Java não desbancou C++ como a linguagem de sistemas nos anos 2000, apesar de sua enorme popularidade na época? Cite ao menos uma razão que torna Java inadequada para programação desses tipos de sistemas computacionais.

   (b) (5 Pontos) COBOL foi, durante as décadas de 1960 e 1970, a principal linguagem de negócios, amplamente utilizada em sistemas bancários e corporativos. Esses programas eram caracterizados por seu grande porte, complexidade, divisão em múltiplos módulos e desenvolvimento por equipes extensas. A partir dos anos 1990, porém, Java (e, em menor grau, C#) passou a ocupar esse espaço, tornando-se a linguagem dominante em sistemas corporativos modernos. Mesmo Python, que hoje é mais popular do que Java, não assumiu esse papel. Por que Python não consegue substituir Java no desenvolvimento desses sistemas corporativos de grande escala?

3. O objetivo desta questão é construir uma agenda de compromissos em Prolog. A base de conhecimento abaixo representa compromissos de uma pessoa ao longo do dia, relacionando o horário à descrição (assuma que somente horas inteiras são usadas):

```
compromisso(9,  aula_de_lp).
compromisso(11, reuniao_do_lab).
compromisso(14, escrever_relatorio).
compromisso(16, ginastica).
```

   (a) (2 Pontos) Defina um predicado `tem_compromisso(H)` que seja verdadeiro quando houver um compromisso cadastrado na hora `H`. Por exemplo, `tem_compromisso(11)` deve ser verdadeiro, enquanto `tem_compromisso(10)` deve ser falso.

   (b) (2 Pontos) Defina um predicado `descricao(Hora, Desc)` que relacione a hora de um compromisso à sua descrição. Por exemplo, a consulta `descricao(11, D)` deve produzir `D = reuniao_do_lab`.

   (c) (2 Pontos) Defina um predicado `range(B, E, Lista)` que produza uma lista contendo todos os inteiros de `B` até `E`, inclusive. Por exemplo, a consulta `range(10, 14, R)` deve gerar `R = [10, 11, 12, 13, 14]`.

   (d) (2 Pontos) Defina um predicado `compromisso_intervalo(B, E, Hora, Desc)` que seja verdadeiro quando houver um compromisso na hora `Hora` e essa hora estiver no intervalo entre `B` e `E`, inclusive. O predicado deve também retornar a descrição `Desc` correspondente ao compromisso. Você pode usar `range`, mesmo que não tenha feito a questão anterior. Por exemplo:

   ```
   ?- compromisso_intervalo(10, 15, Hora, Desc).
   Hora = 11,
   Desc = reuniao_do_lab ;
   Hora = 14,
   Desc = escrever_relatorio ;
   ```

   Fique também à vontade para usar o predicado `member(E, L)` em sua resposta.

   (e) (2 Pontos) Defina um predicado `todos_compromissos(B, E, Descs)` que produza, em `Descs`, a lista contendo as descrições de todos os compromissos agendados entre as horas `B` e `E`. Por exemplo:

   ```
   ?- todos_compromissos(10, 15, Descs).
   Descs = [reuniao_do_lab, escrever_relatorio].
   ```

   Fique à vontade para usar `findall(Pattern, Expr, Accumulator)`.