Prova Final de Linguagens de Programação
- DCC024B -
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. Traga o caderno de provas quando vier à mesa do instrutor.
- 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	Questão 4	Questão 5	Extra 0.5

Questão extra (0.5): porque o Congresso Nacional é formado por duas câmeras legislativas: a Câmara de Deputados e o Senado Federal? Em outras palavras, porque entende-se que somente uma câmara não seria suficiente para legislar o país?

1. Essa questão diz respeito a maneira como a linguagem C separa o escopo de variáveis. Para responder as questões, considere o programa abaixo:

```
1. int main() {
2.   int i = 0;
3.   while (i < 10) {
4.     int i = 20;
5.     i = i - 1;
6.   }
7.   return i;
8. }
```

   (a) (2 Pontos) Em qual linha foi declarada a variável `i` usada no condicional da linha 3?

   (b) (2 Pontos) Em qual linha foi declarada a variável `i` cujo valor é lido na linha 5?

   (c) (2 Pontos) Em qual linha foi declarada a variável `i` que é retornada na linha 7?

   (d) (4 Pontos) O programa acima entra em loop, ou ele retorna algum valor? Caso entre em loop, explique o porquê. Caso ele retorne algum valor, qual valor é retornado?

2. Essa questão refere-se a implementação das funções `malloc` e `free` em C.

   (a) (2 Pontos) Qual é a complexidade asimptótica da função `malloc`? Essa função recebe um inteiro n, e retorna um ponteiro para n *bytes* alocados contiguamente em memória.

   (b) (3 Pontos) Por que a função `malloc` sempre retorna blocos de memória alocados contiguamente?

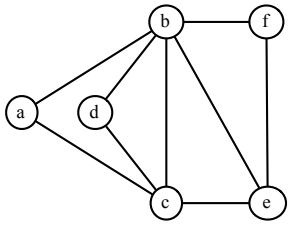
   (c) (2 Pontos) Qual é a complexidade da função `free`? Essa função recebe um ponteiro, e libera a memória apontada por ele.

   (d) (3 Pontos) O programa abaixo imprime 8, em vez de 3. Por que ele imprime 8?

```
int main() {
  int* i = (int*) malloc (sizeof(int));
  int* j;
  *i = 3;
  free(i);
  j = (int*) malloc (sizeof(int));
  *j = 8;
  printf("%d\n", *i);
}
```

3. Dado um grafo $G = (V, E)$, uma *clique* em G é um subgrafo completo de G. Isto é, uma clique é um subgrafo formado por um conjunto $V' \subseteq V$, tal que para quaisquer $v_1 \in V'$ e $v_2 \in V'$, tem-se que a aresta $(v_1, v_2) \in E$. Nesta questão você deverá implementar um predicado em Prolog que encontra cliques de tamanho N. Um grafo é dado por um conjunto de predicados `edge(x, y)`, que indica que os vértices `x` e `y` são conectados. Um exemplo de grafo pode ser visto logo abaixo.

```
edge(a, b).
edge(a, c).
edge(b, c).
edge(c, d).
edge(c, e).
edge(b, d).
edge(b, e).
edge(e, f).
edge(b, f).
```




```
?- cliqueN([a, b, c, d, e, f], C, 3).
C = [a, b, c] ;
C = [b, c, d] ;
C = [b, c, e] ;
C = [b, e, f]
```

(a) (5 Pontos) Escreva um predicado `sublist(L, S)` que seja verdade quando a lista `S` for sublista da lista `L`. Exemplo:

```
?- sublist([a, b], S).
S = [a, b] ;
S = [a] ;
S = [b] ;
S = [].
```

(b) (5 Pontos) Escreva um predicado `cliqueN(V, C, N)`, que seja verdade quando `C` for um subconjunto de tamanho `N` da lista de vértices `V`. Um exemplo de uso deste predicado pode ser visto na figura acima. Para lhe auxiliar, você pode usar os predicados abaixo. Fique livre para escrever quaisquer predicados auxiliares que julgar necessário. Importante: seu predicado deve funcionar para qualquer grafo, não somente para o grafo dado como exemplo na figura acima.

```
linked(X, Y) :- edge(X, Y).
linked(X, Y) :- edge(Y, X).

clique([]).
clique([_]).
clique([X1,X2|R]) :- linked(X1, X2), clique([X1|R]), clique([X2|R]).

length([], 0).
length([_|T], N) :- length(T, NN), N is NN + 1.
```

4. Esta questão refere-se ao programa abaixo, o qual está escrito em Java. Preencha cada espaço com uma das seguintes sentenças, de acordo com a invocação que será feita. Cada acerto vale 1 ponto:

   - "Será invocado método N", sendo $N \in \{1, 2, 3, 4, 5\}$;
   - "Erro de compilação". Em caso de erro de compilação, assuma, para as próximas linhas, que o objeto pôde ser criado normalmente;
   - "Erro de execução".

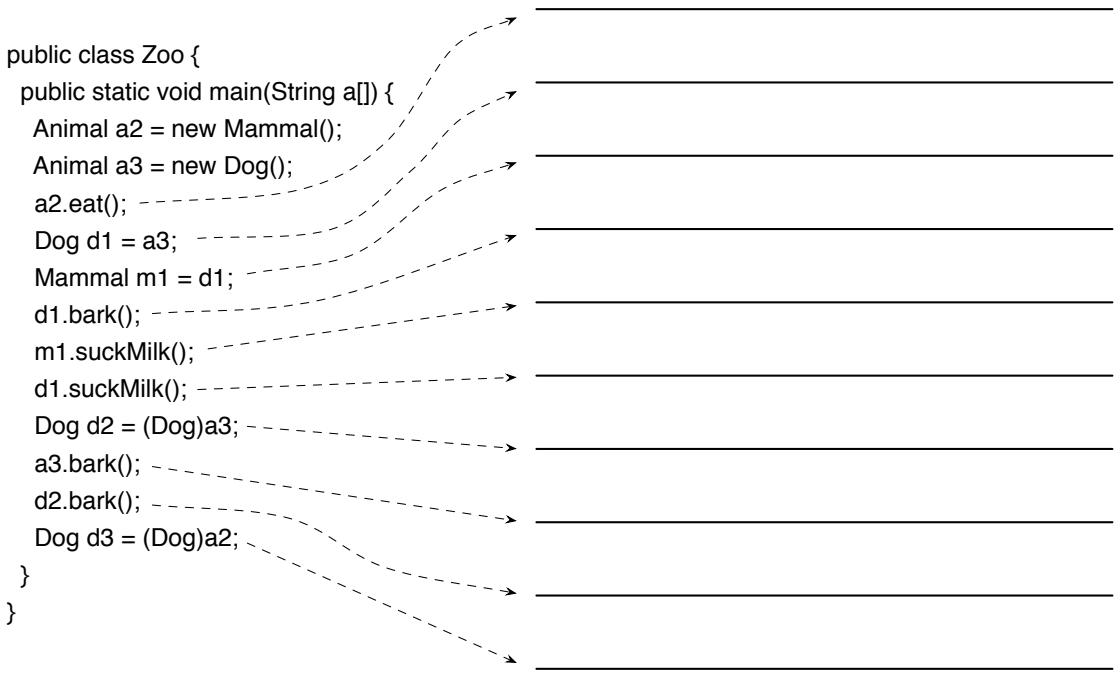
```
public class Animal {
[1] public void eat() { System.out.println("Animal come"); }
}

public class Mammal extends Animal {
[2] public void suckMilk() { System.out.println("Mamífero mama"); }
}

public class Dog extends Mammal {
[3] public void bark() { System.out.println("Cachorro late"); }
[4] public void suckMilk() { System.out.println("Cachorro mama"); }
[5] public void eat() { System.out.println("Cachorro come"); }
}

public class Zoo {
  public static void main(String a[]) {
    Animal a2 = new Mammal();
    Animal a3 = new Dog();
    a2.eat();
    Dog d1 = a3;
    Mammal m1 = d1;
    d1.bark();
    m1.suckMilk();
    d1.suckMilk();
    Dog d2 = (Dog)a3;
    a3.bark();
    d2.bark();
    Dog d3 = (Dog)a2;
  }
}
```

Os métodos que podem ser invocados são esses de 1 a 5.



5. (10 Pontos) A sintaxe de Python provê uma cláusula `else` para laços. Em geral, linguagens da família BCPL (C, C++, Java, C#) não contêm nada semelhante. O programa abaixo ilustra o uso deste tipo de `else`. O que a função `mystery` faz?

```
def mystery(limit):
  nums = []
  for n in range(2, limit):
    for x in range(2, n):
      if n % x == 0:
        break
    else:
      nums.append(n)
  return nums
```