Aluno (a) : __GABARITO_____

1. NÃO é uma vantagem de linguagens estaticamente tipadas:

(a) Pensar em tipos é também uma forma de projetar a arquitetura do software.
(b) Proporcionam um ganho de desempenho, pois eliminam a necessidade de testes de tipo durante a execução do programa.
(c) São amigáveis com ferramentas de programação como ambientes IDE.
(d) Antecipam certos tipos de erros.
(e) Proporcionam uma experiência mais flexível de programação.

2. Observe o seguinte código (parece C, mas não é):

```
int z = 1000
void f(int k) {
  k++;
}
void main() {
  f(z);
  print(z)
}
```

Considere que o programa acima funcionasse ao mesmo tempo para duas linguagens de programação com a sintaxe do C. A semântica de passagem de parâmetros da primeira linguagem seria por referência e a segunda, por valor. Os resultados da execução do programa nas duas linguagens seria respectivamente (impressão):

(a) 1001 e 1000
(b) 1001 e 1001
(c) 1000 e 1000
(d) 1000 e 1001

3. Com respeito a tipos abstratos de dados em linguagens de programação, pode-se afirmar:

(a)`Quando há isolamento entre o cliente e a implementação, esta pode ser trocada.
(b) Ao se abstrair conceitos, nada fica oculto no software.
(c) Tipos abstratos de dados e classes (da orientação a objetos) são a mesma coisa.
(d) O ideal é que o cliente seja independente da especificação.
(e) A representação da interface é a abstração.

B – Abstrair é priorizar certos aspectos em detrimentos de outros, que se pretende desconhecer (ocultar) o máximo possível.
C – A linguagem Scala é um contra-exemplo: ela possui os dois.
D – O programa cliente deve depender apenas da especificação, isolando da implementação.
E – Representação = implementação. Não! A abstração é a interface apenas.

4. Por que a seguinte função NÃO representa o paradigma funcional em linguagens de programação?

```
int g(int a) {
  b++;
  return a;
}
```

(a) Porque nela não se realiza atribuição.
(b) Porque seu resultado só depende do parâmetro.
(c) Porque ela não apresenta efeitos colaterais.
(d) Porque ela não apresenta transparência referencial.
(e) Porque ela não possui sentenças (*statements*).

A – b++ é uma atribuição. Atribuições acontecem em linguagens imperativas.
B – nada a ver.
C – b não é local = efeito colateral.
E – b++ é uma sentença, return a é outra.

5. Preencha com V ou F para indicar se a sentença é verdadeira ou falsa. É possível deixar em branco. Um erro anula a questão inteira.

(F) Um supertipo é sempre compatível com seu subtipo, isto é, um valor do supertipo pode ser usado em qualquer ponto do programa em que é esperado um valor do seu subtipo.
TROQUE DE LUGAR SUBTIPO COM SUPERTIPO
(V) Linguagens de escopamento dinâmico geralmente são dinamicamente tipadas.
(F) Conversão de tipo é uma forma automática de compatiblidade de tipos.
COERÇÃO
(V) Linguagens modernas adotam a equivalência nominal de tipos.

6. Idem ao anterior:

(V) A semântica de passagem de parâmetros por nome coincinde com a semântica de execução *lazy evaluation* em seus efeitos.
(F) Para objetos em Java, a semântica de passagem de parâmetros pode ser tanto por valor quanto referência.
REFERÊNCIA
~~( ) Programação genérica e polimorfismo paramétrico são duas propriedades da linguagem Java.~~
ANULADA
(V) A linguagem C só possui passagem de parâmetros por valor.

7. Idem ao anterior:

(V) Blocos em C proporcionam um efeito de encapsulamento mediante o escopo das variáveis.
(V/F) A linguagem C possui tipos abstratos de dados.
(V) Classes em Java são um exemplo de mecanismo de encapsulamento com controle de acesso.
~~( )Hierarquias de subtipo, também conhecidas como *herança*, estão presentes nas linguagens orientadas a objetos.~~

8. Idem ao anterior:

(F) São exemplos de linguagens funcionais estaticamente tipadas: OCaml, Clojure e Erlang.
CLOJURE ERLANG DINÂMICAS
(F) O paradigma funcional tem suas raízes no conceito de Máquinas de Turing.
CÁLCULO LAMBDA
(V) Scheme é um tipo de Lisp, e influenciou o *design* de JavaScript e Lua.
(V) São exemplos de linguagens funcionais: Scala, Lisp, Haskell e F#.

9. Um programador está acostumado a usar uma função de ordenação de uma biblioteca. Um dia, um colega perguntou a ele qual algoritmo de ordenação era empregado nessa função, e ele se deu conta de que nunca precisou saber. Use esse exemplo para explicar abstração de processos em linguagens de programação.

10. Crie um pequeno programa com erros léxicos, sintáticos e semânticos. Indique os erros e os tipos.