TRƯỜNG ĐH BÁCH KHOA – ĐHQG-HCM KHOA KH&KT MÁY TÍNH	**MÔ PHỎNG ĐỀ CUỐI KÌ 222**	**Học kỳ/năm học**	**2**	**2022-2023**
		Ngày thi		**21/05/2023**
	Môn học	**NGUYÊN LÝ NGÔN NGỮ LẬP TRÌNH**		
	Mã môn học	**CO3005**		
	Thời lượng	**120 phút**	**Mã đề**	**2023**

Ho và tên SV: ……DHHK……MSSV:……Nhóm lớp:…**L02**……

Ghi chú:
- *Sinh viên **ĐƯỢC PHÉP** sử dụng **01 tờ giấy A4 viết tay** trong quá trình làm bài.*
- *Sinh viên **NỘP LẠI ĐỀ THI** sau khi hết giờ làm bài*
- *Đề thi gồm 50 câu, bao gồm 3 câu thuộc khối lập trình (dùng để harmony cho bài tập lớn số 3), 13 câu dạng điền câu trả lời ngắn (dùng để harmony cho bài tập lớn số 4) và 34 câu gồm hai loại: trắc nghiệm và điền câu trả lời ngắn.*
- *Đề thi được chia sẻ dưới sự cho phép của tác giả Đoàn Hồng Hiếu Kiên*

Dùng giả thiết sau để hoàn thành các câu hỏi từ câu 1 đến câu 3.

Cho cây AST của một ngôn ngữ lập trình có kiểu tĩnh và tầm vực tĩnh sau:

```
class AST (ABC)
class Decl (ABC)
class Stmt (ABC)
class Type (ABC)
class Expr (ABC)
class Id (Expr) # name: str
class Program (AST) # decl: List[Decl]
class VarDecl (Decl, Stmt) # name: str, typ: Type
class FuncDecl (Decl) # name: str, param: List[VarDecl], rettype: Type, body: List[Stmt]
class Assign (Stmt) # lhs: Id, rhs: Expr
class CallStmt (Stmt) # name: str, args: List[Expr]
class FuncCall (Expr) # name: str, args: List[Expr]
class BinExpr (Expr) # op: str (+, -, *, /), expr1: Expr, expr2: Expr
class IntLit (Expr) # val: int
class FloatLit (Expr) # val: float
class IntType (Type)
class FloatType (Type)
class VoidType (Type)
```

Cho *o* là một cấu trúc dữ liệu dạng List hai chiều, mỗi phần tử của mỗi sublist trong *o* là một tuple gồm ba thành phần: *name* (kiểu str), *typ* (kiểu Type) và *params* (kiểu List[tuple], mỗi thành phần của tuple này gồm *name* (kiểu str) và *typ* (kiểu Type), với khai báo biến thì *params* được đặt là None).

Class StaticChecker có sẵn một phương thức sau:

```
def lookup(self, attr, lst, func)
```

Phương thức này trả về phần tử đầu tiên trong danh sách `lst` có cùng tính chất với `attr` sau khi đã áp dụng hàm `func` cho phần tử đó. Nếu không có phần tử nào thỏa mãn, trả về None.

Câu 1: Hiện thực các phương thức sau của class StaticChecker:

```
def visitProgram(self, ast: Program, o)
def visitVarDecl(self, ast: VarDecl, o)
def visitFuncDecl(self, ast: FuncDecl, o)
```

Trong quá trình visit, lỗi sau có thể xảy ra:

`Redeclared(kind: Variable | Function | Parameter, name: str)`: Được raise khi StaticChecker phát hiện hai khai báo bất kì có cùng tên trong cùng một tầm vực. Đối với hàm, việc khai báo lại các tham số của hàm bên trong thân hàm là **không** được phép.

Câu 2: Hiện thực các phương thức sau của class StaticChecker:

```
def visitAssign(self, ast: Assign, o)
def visitBinExpr(self, ast: BinExpr, o)
def visitIntLit(self, ast: IntLit, o)
def visitFloatLit(self, ast: FloatLit, o)
```

Biết rằng:

- Trong Assign, nếu vế bên trái có kiểu là float, vế bên phải có thể có kiểu float hoặc integer. Ngược lại thì hai vế phải có cùng kiểu

- Trong BinExpr, nếu một trong hai biểu thức có kiểu là float, cả biểu thức có kiểu là float, ngược lại thì cả biểu thức có kiểu là integer

Các lỗi sau có thể xảy ra:

`Undeclared(name: str)`: Khi có một biến nào đó được sử dụng nhưng không được khai báo ở bất kỳ tầm vực nào. Việc tìm kiếm tên biến được thực hiện từ tầm vực trong cùng, đi ra dần cho đến tầm vực toàn cục

`TypeMismatch(e: Stmt | Expr)`: Khi có một statement hoặc một expression vi phạm về kiểu được quy định ở trên

Câu 3: Hiện thực các phương thức sau của class StaticChecker:

```
def visitCallStmt(self, ast: CallStmt, o)
def visitFuncCall(self, ast: FuncCall, o)
```

Biết rằng:

- Trong CallStmt, hàm phải có kiểu trả về là void. Danh sách đối số args của CallStmt phải tương thích với hàm đã được khai báo trước đó (cùng số lượng tham số, kiểu của các đối số tuân theo quy tắc: nếu tham số có kiểu là float, đối số truyền vào tương ứng có thể có kiểu là integer hoặc float, ngược lại thì tham số và đối số truyền vào phải có cùng kiểu)
- Trong FuncCall, hàm phải có kiểu trả về khác void. Danh sách đối số args tuân theo quy tắc giống với CallStmt

Các lỗi sau có thể xảy ra:

`Undeclared(name: str)`: Khi có một hàm nào đó được sử dụng nhưng không được khai báo ở tầm vực toàn cục (hàm chỉ được khai báo ở tầm vực toàn cục và không cho phép sử dụng hàm trước khi hàm được khai báo)

`TypeMismatch(e: Stmt | Expr)`: Khi có một statement hoặc một expression vi phạm về kiểu được quy định ở trên. Nếu đối số truyền vào hàm là BinExpr, quy tắc về BinExpr được nêu ở câu hỏi 2 sẽ được áp dụng

Dùng giả thiết sau để hoàn thành các câu hỏi từ câu 4 đến câu 16

Trong ngôn ngữ lập trình X, vòng lặp For có cấu trúc như sau:

```
for <scala-variable> until <condition-expr> by <update-expr>
      <statement>
else <statement>
```

Cách hoạt động như sau: đầu tiên giá trị của `scala-variable` sẽ được sử dụng để bắt đầu một vòng lặp và biểu thức `<condition-expr>` sẽ được kiểm tra, nếu biểu thức này trả về false, lệnh `<statement>` sẽ được chạy. Sau đó cập nhật giá trị mới của `scala-variable` bằng cách cộng thêm `<update-expr>` vào giá trị hiện tại của `<scala-variable>`. Vòng lặp này cứ tiếp diễn cho đến khi biểu thức `<condition-expr>` trả về true, khi đó vòng lặp dừng lại và câu lệnh `<statement>` sau từ khóa `else` sẽ được thực thi.

Cho khai báo các class sau:

```
class Type
class MType # partype: List[Type], rettype: Type
class Val
class CName (Val) # value: str
class Index (Val) # value: int
class Symbol # name: str, mtype: MType | Type, value: Val
class SubBody # frame: Frame, sym: List[List[Symbol]]
class CodeGenerator # emit: Emitter
```

Các phương thức được sử dụng trong class Frame:

```
def enterLoop(self) -> None
def getNewLabel(self) -> int
def getContinueLabel(self) -> int
def getBreakLabel(self) -> int
def exitLoop(self) -> None
```

Các phương thức được sử dụng trong class Emitter:

```
def emitLABEL(self, label: int, frame: Frame) -> str
def emitIFTRUE(self, label: int, frame: Frame) -> str
def emitIFFALSE(self, label: int, frame: Frame) -> str
def emitGOTO(self, label: int, frame: Frame) -> str
def printout(self, code: str) -> None
```

Điền vào chỗ trống các lệnh còn thiếu để sinh mã Java bytecode cho vòng lặp for của ngôn ngữ lập trình X (chỉ số dòng lệnh bắt đầu từ 1).

```
def visitFor(self, ast: For, o: SubBody):
```

```
        frame = o.frame
        frame.enterLoop()
        forLabel, elseLabel = # 4: TODO
        conLabel, brkLabel = # 5: TODO
        scalaCode = self.visit(ast.name, o)[0]
        # 6: TODO – printout the code for scala variable
        # 7: TODO - put the label to start one iteration
        expCode = self.visit(ast.condExpr, o)[0]
        # 8: TODO – print the code for condition expression
        # 9: TODO – jump to elseStmt if condition is true
        self.visit(ast.body, o)
        # 10: TODO – put label for update scala variable here
        self.visit(AssignStmt(ast.name, BinExpr('+', ast.name, ast.updExpr)) # generate Java
bytecode for updating scala variable, count this code and comment as 1 line of code
        # 11: TODO – end current iteration, jump to next iteration
        # 12: TODO – put label for else statement
        self.visit(ast.elseStmt, o)
        # 13: TODO – put the exit label here
        frame.exitLoop()
```

Câu 14: Có bao nhiêu câu lệnh Java bytecode được sinh ra sau khi thực hiện lệnh ở dòng thứ 14 (giả sử i là local variable có vị trí trong local variable array là 1) cho vòng lặp for sau ?

```
for i until i > 10 by 1
        <statement>
else <statement>
```

Câu 15: Xác định kích thước của operand stack sau khi thực hiện đoạn code trên với vòng lặp for dưới đây:

```
for i until i > 10 by 1
        i <- i + 3
else writeNumber(i)
```

Câu 16: Giả sử sau khi kết thúc vòng lặp và trước khi thực thi câu lệnh sau từ khóa `else`, giá trị của `<scala-variable>` sẽ được khôi phục lại thành giá trị của `<scala-variable>` trước khi thực hiện vòng lặp for. Khi đó đoạn code trên cần phải thay đổi những dòng lệnh nào?
Note: Nếu muốn thêm một khối lệnh sau dòng lệnh thứ i, điền chỉ số thứ i + 1. Nếu muốn xóa dòng lệnh thứ i, điền chỉ số thứ i, không cho phép xóa một khối lệnh. Các chỉ số dòng lệnh được ngăn cách nhau bởi dấu phẩy và không có bất kỳ khoảng trắng nào.

Câu 17: Chuyển đổi số thập phân 3.59 sang dạng nhị phân cho một máy tính X theo chuẩn IEEE – 754, biết rằng dạng nhị phân của máy tính X này gồm có 8 bit, với 1 bit dấu, 3 bit cho phần mũ và 4 bit cho phần thập phân

Câu 18: Cho cấu trúc dữ liệu sau trong ngôn ngữ X:

```
record {
        x: integer;
        y: record {
                z: array [1...10] of ^ real;
                t: real;
        };
}
```

Trong đó ^ là ký hiệu cho kiểu dữ liệu con trỏ
Viết biểu thức kiểu mô tả kiểu cho cấu trúc dữ liệu trên, với số lượng biến kiểu là ít nhất có thể
Note: sử dụng `*` thay cho ×, `->` thay cho →, chỉ sử dụng các kiểu dữ liệu cơ bản (integer, real) và pointer(type) nếu có xuất hiện kiểu dữ liệu con trỏ, kết quả cuối cùng không có bất kỳ khoảng trắng nào.

Structure data type sau trong C được sử dụng cho các câu hỏi từ 19 đến 20:

```
struct X {
        int x;
        char y;
        bool z;
        double t;
```

```
        float* e;
};
```
Biết rằng trình biên dịch ngôn ngữ C dùng để biên dịch kiểu dữ liệu trên được cài đặt trên máy tính có cấu hình vi xử lý 32 bit.

Câu 19: Xác định kích thước của kiểu dữ liệu X

Câu 20: Xác định kích thước nhỏ nhất có thể của kiểu dữ liệu X khi thay đổi thứ tự các thuộc tính có trong X

Đoạn chương trình sau được sử dụng cho các câu hỏi từ câu 21 đến câu 23:
```
var a, b: string(6) // create two variables with type string with maximum length of 6
a ≔ ‘toi’
b ≔ ‘ve nha’
writeString(a + b)
```
Với phép + là phép nối hai chuỗi

Câu 21: Giả sử kiểu string trong ngôn ngữ trên được hiện thực ở dạng tĩnh (static). Kết quả được in ra màn hình là gì?

Câu 22: Cũng như câu hỏi 21 nhưng kiểu string được hiện thực ở dạng động có giới hạn (limit dynamic)

Câu 23: Cũng như câu hỏi 21 nhưng kiểu string được hiện thực ở dạng động hoàn toàn (dynamic)

Câu 24: Giả sử ngôn ngữ X có kiểu set được lưu trữ ở dạng chuỗi bit (bit chain) và khai báo sau dùng để khởi tạo một biến có kiểu tập hợp gồm các số nguyên từ 1 đến 128:
```
Var x: set of 1...128
```
Kích thước của biến x là bao nhiêu?

A. 8 bytes B. 16 bytes C. 128 bytes D. 7 bytes

Câu 25: Sinh mã Java bytecode cho câu lệnh sau: `b[3] = a`, biết rằng a, b là hai local variable có chỉ số trong local variable array tương ứng là 1 và 3, trong đó b là mảng một chiều có kiểu phần tử là integer còn a có kiểu integer

Câu 26: Cho đoạn chương trình của ngôn ngữ Python như sau:
```
for i in range(10):
        i += 3
print(i)
```

Mã giả (pseudo – code) của vòng lặp for trên như sau:
```
i = iterable_obj[0]
Label 0:
        if i is None: goto Label 1
        <some statement>
        i = next(iterable_obj) # get the next item in iterable_obj, returns None if current
item is at the end of iterable_obj
        goto Label 0
Label 1:
        <some statement>
```

Xác định giá trị được in ra màn hình sau khi thực hiện lệnh dòng 3 (ghi nhớ rằng Python sử dụng cơ chế dynamic binding, do đó lệnh ở dòng 3 vẫn hợp lệ và không có lỗi NameError)

A. 10 B. 9 C. 12 D. 13

Đoạn chương trình sau được sử dụng cho các câu hỏi từ câu 27 đến câu 29:
```
foo: function void (int a, int b, int c) {
        a = a + c;
        b = a + b + c;
        print a, b, c;
}
```

```
main: function void () {
	int j = 10, j = 15;
	foo (j, k, j + k);
	print k, j;
}
```

Biết rằng trình biên dịch sẽ thực thi hàm main đầu tiên sau khi chương trình được biên dịch thành công

Câu 27: Các tham số trong hàm foo được truyền theo quy tắc: a, b được truyền tham khảo, c được truyền giá trị. Kết quả khi in ra màn hình là?

Câu 28: Giả sử a được truyền tham khảo, b được truyền giá trị - kết quả, c được truyền giá trị. Kết quả khi in ra màn hình là?

Câu 29: Chương trình có được thực thi thành công hay không nếu c được truyền giá trị - kết quả?

Đoạn chương trình sau được sử dụng cho các câu hỏi từ câu 30 đến câu 33:

```
function f1()
{
    var x = 10;
    function f2(fx)
    {
        var x;
        x = 6;
        fx();
    };

    function f3()
    {
        print x;
    };

    f2(f3);
};
```

Câu 30: Giả sử ngôn ngữ được chọn là ngôn ngữ tầm vực tĩnh. Xác định kết quả được in ra màn hình sau khi thực thi hàm `f1()`

Câu 31: Giả sử ngôn ngữ được chọn là ngôn ngữ tầm vực động, xác định môi trường tham khảo tên của `f3` khi thực thi chương trình trên theo thứ tự: `f1 -> f2 -> f3`

Câu 32: Giả sử ngôn ngữ được chọn là ngôn ngữ tầm vực động sử dụng cơ chế shallow binding. Xác định kết quả được in ra màn hình sau khi thực thi hàm `f1()`

Câu 33: Cũng yêu cầu như câu 32 nhưng ngôn ngữ được chọn là ngôn ngữ tầm vực động sử dụng cơ chế deep binding

Đoạn chương trình sau trong ngôn ngữ C++ được sử dụng cho các câu hỏi từ câu 34 đến câu 38:

```
int* m;
int* foo (int x) {
	static int y;
	int* z = new int (x);
	switch x {
		case 1: return &y;
		case 2: return &x;
		case 3: return z;
		default: return m;
	}
}
```

Đối với các câu hỏi từ câu 34 đến câu 37, chỉ điền duy nhất 1 chữ cái vào ô trống, hoặc **new int** (7 kí tự)

Câu 34: Đối tượng ____ có thời gian sống bằng với thời gian sống của biến z

Câu 35: Đối tượng ____ có thời gian sống bằng với thời gian thực thi của hàm foo

Câu 36: Đối tượng ____ có thời gian sống bằng với biến m

Câu 37: Đối tượng ____ có thời gian sống dài hơn thời gian thực thi của hàm foo nhưng không bằng thời gian thực thi của chương trình chính

Câu 38: Đoạn chương trình trên có bị lỗi gì hay không khi thực thi foo (2) ?

A. Tham chiếu treo (Dangling reference)

B. Tạo ra rác (Garbage)

C. Gây ra alias

D. Không có lỗi gì cả

Đoạn chương trình sau trong ngôn ngữ C++ được sử dụng cho các câu hỏi từ câu 39 đến câu 40:

```
int s = 0;
int* p = &s
int* q = p
int r = &s
```

Câu 39: Xác định các alias được tạo ra trong đoạn chương trình trên (các alias được viết theo thứ tự alphabet, ngăn cách nhau bằng dấu phẩy và không chứa khoảng trắng. Nếu có sử dụng kí hiệu * hoặc & thì viết các kí hiệu này liền trước ký tự, ưu tiên * trước rồi đến & nếu trùng tên)

Câu 40: Thực hiện gán `*q = 200`. Xác định giá trị của `s, *p, r`

Dùng đoạn chương trình sau cho các câu hỏi từ câu 41 đến câu 45 (chỉ số dòng lệnh bắt đầu từ 1)

```
var x, y: integer;
func foo1 (x: integer)
begin
	var z: integer;
	func foo2 (y: integer)
	begin
		var z: float;
		func foo3 (x: integer, y: float)
		begin … end
		func foo4 ()
		begin … end
	end
end
```

Câu 41: Môi trường tham khảo tên của foo3 sử dụng ngôn ngữ tầm vực tĩnh **không** chứa đối tượng nào sau đây?

A. z ở dòng 7 B. x ở dòng 8 C. y ở dòng 8 D. z ở dòng 4

Câu 42: Trong foo2, danh sách tất cả các tên có thể truy cập sử dụng ngôn ngữ tầm vực tĩnh là?

A. x//2, z//4, y//5, foo1, foo2, foo3, foo4

B. x//1, y//1, x//2, y//5, foo1, foo2, foo3, foo4

C. x//2, y//4, z//6, foo1, foo2, foo3, foo4

D. x//8, y//8, z//7, foo1, foo2, foo3, foo4

Câu 43: Giả sử chương trình được thực thi theo thứ tự: main gọi foo1, foo1 gọi foo2, foo2 gọi foo3, foo3 gọi foo4. Trong quá trình thực thi, các đối tượng sau được tạo ra:

main: x(o1), y(o2)

foo1: x(o3), z(o4)

foo2: y(o5), z(o6)

foo3: x(o7), y(o8)

Môi trường tham khảo động của foo4 bao gồm những đối tượng nào?

A. o7, o8, o4 B. o5, o6, o3 C. o7, o8, o6 D. o1, o2, o4

Câu 44: Giả sử chương trình được thực thi theo thứ tự: main gọi foo1, foo1 gọi foo2, foo2 gọi đệ quy một lần, foo2 gọi foo3, foo3 gọi foo4. Trong quá trình thực thi, các đối tượng sau được tạo ra:
main: x(o1), y(o2)
foo1: x(o3), z(o4)
foo2_1: y(o5), z(o6)
foo2_2: y(o7), z(o8)
foo3: x(o9), y(o10)
Môi trường tham khảo động của foo3 bao gồm những đối tượng nào?

A. o9, o7, o8 B. o9, o7, o6 C. o9, o10, o8 D. o5, o6, o9

Câu 45: Xác định môi trường tham khảo tên của foo4 sử dụng ngôn ngữ tầm vực tĩnh? (Các tên được viết theo thứ tự alphabet, ngăn cách nhau bởi dấu phẩy và không có khoảng trắng. Nếu có nhiều dòng lệnh có trùng tên, viết theo thứ tự: tên//chỉ số dòng lệnh)

Câu 46: Sinh mã Java bytecode cho câu lệnh sau: b = a + 2.0
Với a có kiểu là integer và b có kiểu là float, a và b là hai local variable có chỉ số trong local variable array tương ứng là 0 và 1

Câu 47: Chuyển đổi biểu thức trung tố sau sử dụng Polish Notation, Cambridge Polish Notation và Reverse Polish Notation: (1 + 2) × (3 + 4) / 7

Câu 48: Cơ chế nào giúp cho đoạn chương trình sau không bị lỗi khi thực thi?

```
int x = 20;
if (x >= 0 || 1 / 0 == 0) {
	<some statement>
}
else {
	<some statement>
}
```

A. Thứ tự ưu tiên tính toán của toán tử
B. Short – circuit evaluation
C. Do người làm compiler xử lý exception
D. Các câu trả lời trên đều sai

Đoạn chương trình sau được sử dụng cho các câu hỏi từ câu 49 đến câu 50

```
abstract class A {
	abstract void foo1() {}
	abstract void foo2() {}
}
class B extends A {
	void foo1() {}
}
class C extends A {
	void foo1() {}
	void foo2() {}
}
class D extends B {
	void foo2() {}
}
```

Câu 49: Khởi tạo con trỏ b có kiểu B và tham chiếu đến các đối tượng, kết luận nào sau đây đúng?
A. b tham chiếu đến đối tượng kiểu A và b.foo1() gọi phương thức foo1 của B
B. b tham chiếu đến đối tượng kiểu B và b.foo2() gọi phương thức foo2 của A
C. b tham chiếu đến đối tượng kiểu C và b.foo1() gọi phương thức foo1 của C
D. b tham chiếu đến đối tượng kiểu D và b.foo1() gọi phương thức foo1 của B

Câu 50: Có bao nhiêu lớp trừu tượng (abstract class) được tạo ra trong đoạn chương trình trên?

A. 1 B. 2 C. 3 D. 4

HẾT