KHOA **KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH** Họ và tên:.....
BỘ MÔN **KHOA HỌC MÁY TÍNH** MSSV:.....

Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình
Thời gian: 120 phút Ngày thi: 22-12-2017

☐ Sinh viên được phép sử dụng tài liệu ☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1171

- Sinh viên phải ghi tên và mã số sinh viên trên đề thi và giấy làm bài. Khi nộp bài, sinh viên phải nộp cả **đề thi và giấy làm bài**. Nếu sinh viên không nộp lại đề thi thì sẽ bị kỷ luật cấm thi theo qui chế học vụ (24.1.c).

I. Phần câu hỏi lập trình (đồng thời dùng tính điểm bài tập lớn): (2 điểm)

1. Viết phương thức **visitFor(ast:For,o:Context)** để sinh mã cho một phát biểu For. Cho biết lớp của phát biểu For trên AST được định nghĩa như sau:

   case class For(val expr1: Expr, val expr2: Expr, val expr3: Expr, val loop: Stmt) extends Stmt

   Một số phương thức của Emitter có thể sử dụng:

   - emitIFTRUE(label: Int, frame: Frame)
   - emitIFFALSE(label: Int, frame: Frame)
   - emitGOTO(label: Int, frame: Frame)
   - emitLABEL(label: Int, frame: Frame)
   - emitPUSHICONST(in: Int, frame: Frame)
   - emitANDOP(frame: Frame)
   - emitOROP(frame: Frame)
   - printout(str: String)

   Một số phương thức của Frame có thể sử dụng:

   - getNewLabel()
   - getBreakLabel()
   - getContLabel()
   - getStartLabel()
   - getEndLabel()

2. Giả sử chỉ có các phép toán AND (&&) trên biểu thức nhị phân, hãy viết **visitBinaryOp(ast:BinaryOp, o:Context)** để sinh mã cho các phép toán AND. Chú ý phải sinh mã cho phép rút ngắn tính toán (short-circuit) thì mới được trọn điểm câu này. Nhắc lại khai báo lớp BinaryOp trên cây AST như sau:

   case class BinaryOp(op:String,left:Expr,right:Expr) extends Expr

   Các phương thức của Emitter và Frame đã nêu ở câu trên.

II. Phần câu hỏi tự luận:(8 điểm)

3. Cho một cấu trúc dữ liệu được định nghĩa trên Scala như sau:

```
trait Element
case class Many(value:List[Element]) extends Element
case class Inte(value:Int) extends Element
case class Flt(value:Float) extends Element
```

và hàm **sum(lst:List[T],f:T=>Int):Int** trả về tổng các giá trị nguyên được ánh xạ từ các phần tử của danh sách lst qua hàm f. Ví dụ sử dụng hàm sum để tính tổng của một danh sách: sum(List(1,3,4),(x:Int)=>x) sẽ trả về 8.

Hãy viết lệnh gọi hàm sum trên để đếm số phần tử Inte trực tiếp (phần tử của danh sách) và gián tiếp (đệ qui trong các phần tử của danh sách) trong một danh sách các Element? Ví dụ:
val lst = List(Many(List(Inte(3),Many(List(Flt(2.1),Inte(5))),Flt(1.0),Inte(2))),Inte(1))
và sum(lst,...) sẽ trả về 4, với ... là nội dung cần phải thực hiện cho câu này.

4. Cho đoạn mã sau được viết trên ngôn ngữ Scala dùng **qui tắc tầm vực tĩnh (static-scope rule)**

```
def main = {
    var a = 0; var b = 1; var c = 2 //1
    def sub1(a:Int) = {                //2
        def sub2(a:Int,c:Int,f:Int=>Int) = (f(a) - f(c))/2   // 3
        def sub3(b:Int) = b * c + a; // 4
        b = sub2(0,1,sub3)
    }
    sub1(3)
    print(b) // 5
}
```

   (a) Cho biết môi trường tham khảo tĩnh (static referencing environment) của các hàm **main, sub1, sub2, sub3**.

   (b) Hãy vẽ các bản ghi hoạt động của các chương trình con còn đang tồn tại khi hàm main được gọi và chương trình thực thi đến dòng lệnh trong thân hàm sub3 **// 4** lần thứ nhất? Cho biết kết quả in giá trị b của bản hoạt động main ở dòng // 5 là bao nhiêu?

5. Hãy nêu các đặc điểm chính của cơ chế Gọi - Trở về đơn giản? Đối với mỗi đặc điểm, hãy cho biết cơ chế gọi chương trình con nào khác biệt với cơ chế này?

6. Hãy giải thích các phương pháp (lock-and-key, tombstone) tránh tham chiếu treo (dangling reference)? Nêu rõ cách truy xuất trong trường hợp bình thường, khi huỷ bỏ một đối tượng và cách phát hiện khi có tham chiếu treo.

7. Hãy viết lại biểu thức ở dạng trung tố (infix) sau sang dạng biểu thức tiền tố (prefix) Cambridge Polish:

   a - b * c * d - e + f

   Biết rằng độ ưu tiên và tính kết hợp của các phép toán trong biểu thức như thông thường (đều kết hợp trái và * có ưu tiên cao hơn +, -). Trong biểu thức tiền tố nhiều toán hạng, thứ tự tính toán cũng từ trái sang phải.

   Yêu cầu: Biểu thức dạng tiền tố Cambridge Polish phải thỏa các yêu cầu sau:

- Thứ tự xuất hiện các toán hạng trong biểu thức dạng tiền tố phải có cùng thứ tự xuất hiện như trong biểu thức trung tố.
- Số (và) là ít nhất.
- Có cùng thứ tự tính toán các phép toán với thứ tự đó trong biểu thức dạng trung tố.

8. Giải thích đặc điểm của kiểu union? Cho một ví dụ sử dụng kiểu union và giải thích vì sao phải dùng kiểu union trong ví dụ của em?

9. Hãy thực hiện (viết các phương trình thể hiện các ràng buộc kiểu) suy diễn kiểu để suy ra kiểu của hàm sau:

```
H(x,f,h) {
        if (f(f(x))) return h(x); else return f(x);
}
```

Các ràng buộc kiểu trên các phát biểu if tương tự như trên MC. Kiểu của biểu thức sau **return** phải cùng kiểu trả về của hàm.

——————————————————HẾT——————————————————

Chủ nhiệm bộ môn Chữ kí: Họ tên:	Giảng viên ra đề Chữ kí: Họ tên:

KHOA **KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH** Họ và tên:.....
BỘ MÔN **KHOA HỌC MÁY TÍNH** MSSV:.....

Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình
Thời gian: 120 phút Ngày thi: 22-12-2017

☐ Sinh viên được phép sử dụng tài liệu ☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1172

- Sinh viên phải ghi tên và mã số sinh viên trên đề thi và giấy làm bài. Khi nộp bài, sinh viên phải nộp cả **đề thi và giấy làm bài**. Nếu sinh viên không nộp lại đề thi thì sẽ bị kỷ luật cấm thi theo qui chế học vụ (24.1.c).

I. Phần câu hỏi lập trình (đồng thời dùng tính điểm bài tập lớn): (2 điểm)

1. Viết phương thức **visitFor(ast:For,o:Context)** để sinh mã cho một phát biểu For. Cho biết lớp của phát biểu For trên AST được định nghĩa như sau:

   case class For(val expr1: Expr, val expr2: Expr, val expr3: Expr, val loop: Stmt) extends Stmt

   Một số phương thức của Emitter có thể sử dụng:

   - emitIFTRUE(label: Int, frame: Frame)
   - emitIFFALSE(label: Int, frame: Frame)
   - emitGOTO(label: Int, frame: Frame)
   - emitLABEL(label: Int, frame: Frame)
   - emitPUSHICONST(in: Int, frame: Frame)
   - emitANDOP(frame: Frame)
   - emitOROP(frame: Frame)
   - printout(str: String)

   Một số phương thức của Frame có thể sử dụng:

   - getNewLabel()
   - getBreakLabel()
   - getContLabel()
   - getStartLabel()
   - getEndLabel()

2. Giả sử chỉ có các phép toán OR (||) trên biểu thức nhị phân, hãy viết **visitBinaryOp(ast:BinaryOp, o:Context)** để sinh mã cho các phép toán OR. Chú ý phải sinh mã cho phép rút ngắn tính toán (short-circuit) thì mới được trọn điểm câu này. Nhắc lại khai báo lớp BinaryOp trên cây AST như sau:

   case class BinaryOp(op:String,left:Expr,right:Expr) extends Expr

   Các phương thức của Emitter và Frame đã nêu ở câu trên.

II. Phần câu hỏi tự luận:(8 điểm)

3. Hãy viết lại biểu thức ở dạng trung tố (infix) sau sang dạng biểu thức tiền tố (prefix) Cambridge Polish:

   a - b * c - d * e + f

   Biết rằng độ ưu tiên và tính kết hợp của các phép toán trong biểu thức như thông thường (đều kết hợp trái và * có ưu tiên cao hơn +, -). Trong biểu thức tiền tố nhiều toán hạng, thứ tự tính toán cũng từ trái sang phải.

   Yêu cầu: Biểu thức dạng tiền tố Cambridge Polish phải thỏa các yêu cầu sau:

   - Thứ tự xuất hiện các toán hạng trong biểu thức dạng tiền tố phải có cùng thứ tự xuất hiện như trong biểu thức trung tố.
   - Số (và) là ít nhất.
   - Có cùng thứ tự tính toán các phép toán với thứ tự đó trong biểu thức dạng trung tố.

4. Hãy giải thích các phương pháp (lock-and-key, tombstone) tránh tham chiếu treo (dangling reference)? Nêu rõ cách truy xuất trong trường hợp bình thường, khi huỷ bỏ một đối tượng và cách phát hiện khi có tham chiếu treo.

5. Hãy thực hiện (viết các phương trình thể hiện các ràng buộc kiểu) suy diễn kiểu để suy ra kiểu của hàm sau:

```
H(x,f,h) {
        if (x == f(x)) return h(x); else return f(x);
}
```

   Các ràng buộc kiểu trên các phát biểu if, phép toán == tương tự như trên MC. Kiểu của biểu thức sau **return** phải cùng kiểu trả về của hàm.

6. Cho một cấu trúc dữ liệu được định nghĩa trên Scala như sau:

```
trait Element
case class Many(value:List[Element]) extends Element
case class Inte(value:Int) extends Element
case class Flt(value:Float) extends Element
```

   và hàm **sum(lst:List[T],f:T=>Int):Int** trả về tổng các giá trị nguyên được ánh xạ từ các phần tử của danh sách lst qua hàm f. Ví dụ sử dụng hàm sum để tính tổng của một danh sách: sum(List(1,3,4),(x:Int)=>x) sẽ trả về 8.

   Hãy viết lệnh gọi hàm sum trên để đếm số phần tử Flt trực tiếp (phần tử của danh sách) và gián tiếp (đệ qui trong các phần tử của danh sách) trong một danh sách các Element? Ví dụ:
   val lst = List(Many(List(Flt(2.0),Many(List(Flt(2.1),Inte(5))),Flt(1.0),Inte(2))),Flt(1.1))
   và sum(lst,...) sẽ trả về 4, với ... là nội dung cần phải thực hiện cho câu này.

7. Hãy nêu các đặc điểm chính của cơ chế Gọi - Trở về đơn giản? Đối với mỗi đặc điểm, hãy cho biết cơ chế gọi chương trình con nào khác biệt với cơ chế này?

8. Giải thích đặc điểm của kiểu union? Cho một ví dụ sử dụng kiểu union và giải thích vì sao phải dùng kiểu union trong ví dụ của em?

9. Cho đoạn mã sau được viết trên ngôn ngữ Scala dùng **qui tắc tầm vực tĩnh (static-scope rule)**

```
def main = {
    var a = 0; var b = 1; var c = 2 //1
    def sub1(a:Int) = {                //2
        def sub2(a:Int, c:Int,f:Int=>Int) = (f(a) - f(c))/2   // 3
        def sub3(b:Int) = a - b * c // 4
        b = sub2(0,1,sub3)
    }
    sub1(5)
    print(b) // 5
}
```

(a) Cho biết môi trường tham khảo tĩnh (static referencing environment) của các hàm **main, sub1, sub2, sub3**.

(b) Hãy vẽ các bản ghi hoạt động của các chương trình con còn đang tồn tại khi hàm main được gọi và chương trình thực thi đến dòng lệnh trong thân hàm sub3 **// 4** lần thứ nhất? Cho biết kết quả in giá trị b của bản hoạt động main ở dòng // 5 là bao nhiêu?

HẾT

Chủ nhiệm bộ môn Chữ kí: Họ tên:	Giảng viên ra đề Chữ kí: Họ tên: