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: 20-12-2018

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

Mã đề: 1181

- 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. (L0.3) Viết phương thức **visitDowhile(ast:Dowhile,o:Context)** và phương thức **visitContinue(ast:Continue,o:Context)** để sinh mã Jasmin cho một phát biểu Dowhile và Continue; và giải thích làm sao một phát biểu **continue** ở sâu bên trong thân của phát biểu **Dowhile** có thể chuyển điều khiển đến vị trí thích hợp. Nhắc lại, phát biểu do while sẽ thực thi lặp các phát biểu giữa do và while (trong body)cho đến khi điều kiện sau while (expr)trở nên sai. Cho biết lớp của phát biểu Dowhile và Continue trên AST được định nghĩa như sau:

```
class Dowhile(Stmt):
        def __init__(self,body:list(Stmt),expr: Expr)
class Continue(Stmt):
```

Một số phương thức của Emitter có thể sử dụng (nhưng không giới hạn):

- emitIFTRUE(self, label: Int, frame)
- emitIFFALSE(self, label: Int, frame)
- emitGOTO(self, label: Int, frame)
- emitLABEL(self, label: Int, frame)
- emitPUSHICONST(self, in: Int, frame)
- emitANDOP(self, frame)
- emitADDOP(self, op, typ,frame)
- emitMULOP(self, op, typ,frame)

Một số phương thức của Frame có thể sử dụng(nhưng không giới hạn):

- enterLoop(self)
- exitLoop(self)
- getNewLabel(self)
- getBreakLabel(self)
- getContLabel(self)
- getStartLabel(self)
- getEndLabel(self)

2. (LO.3) Giả sử biểu thức nhị phân có các phép toán +, /, AND, trong đó, các phép toán +, / có thể có các toán hạng ở kiểu IntType hoặc FloatType, trong khi phép toán AND chỉ cho phép các toán hạng ở kiểu BoolType. Khi các toán hạng khác kiểu, phải sinh mã chuyển đổi sang kiểu FloatType. Kết quả của phép + và AND có cùng kiểu toán hạng, trong khi phép / luôn trả về kiểu FloatType. Hãy viết **visitBinaryOp(ast:BinaryOp, o:Context)** để sinh mã Jasmin cho biểu thức nhị phân. Chú ý phải sinh mã cho phép **rút ngắn tính toán (short-circuit)** cho AND thì mới được trọn điểm câu này. Nhắc lại khai báo lớp BinaryOp trên cây AST như sau:

```
class BinaryOp(Expr):
        def __init__(self,op:String,left:Expr,right: Expr)
```

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

II. Phần câu hỏi tự luận:(8 điểm)

Pointers refer to an address, references refer toobject or value

3. (LO.2) Hãy trình bày về kiểu con trỏ (pointer) và kiểu tham khảo (reference)? Cho ví dụ cho mỗi kiểu? Nêu các điểm khác biệt của hai loại kiểu này? Hãy giải thích vì sao các loại kiểu này gây ra hiện tượng alias?

4. (LO.2) Hãy nêu điểm khác biệt chính của các cơ chế gọi chương trình con: đệ qui (recursive), biến cố (exception), trình cộng hành (coroutine), trình định thời (scheduled subroutine) và công tác (task) so với cơ chế Gọi-Trở về đơn giản (simple call-return)?

5. (LO.2) 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. Locks-and-keys: Pointer values are represented as (key,address) pairs

6. (LO.2) Hãy thực hiện (viết các phương trình thể hiện các ràng buộc kiểu) suy diễn kiểu để suy ra kiểu của hàm sau:

Tombstone: extra heap cell that is a pointer to the head-dynamic variable

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

Biểu thức điều kiện của phát biểu if phải có kiểu **boolean**. Kiểu của biểu thức sau **return** phải cùng kiểu trả về của hàm.

7. (LO.2) Cho đoạn mã sau được viết trên ngôn ngữ Scala dùng **qui tắc tầm vực tĩnh (static-scope rule)**. Nhắc lại, trên Scala, từ khoá **var** để khai báo biến, **def** để khai báo hàm, Int=>Int là kiểu hàm nhận vào 1 giá trị nguyên trả về 1 giá trị nguyên, giá trị của biểu thức cuối cùng trong một hàm là giá trị trả về của hàm.

```
def main = {
    var a = 0; var b = 1; var c = 4 //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(1,2,sub3)
    }
    sub1(3)
    print(b) // 5
}
```

   (a) Cho biết môi trường tham khảo tĩnh (static referencing environment) của các hàm **main, sub1, sub2, sub3** (với các tên a, b và c phải ghi kèm // dòng khai báo của tên, ví dụ a//1).

   (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? Trình bày tiếp sự thực thi cho đến khi in giá trị b của bản hoạt động main ở dòng // 5?

8. (LO.2) Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:

```
int A[3] = {4,6,14};// index of A starts from 0
int j = 0;
int n = 3;
int sumAndDecrease(int a, int i) {
      int s = 0;
      for ( ; i < n; i = i + 1) {
         s = s + a;
         A[j] = A[j] - 1;
      }
      return s;
}
void main(){
      int s = sumAndDecrease(A[j],j);
      cout << s << A[0] << A[1] << A[2];//1
}
```

Hãy cho biết và giải thích kết quả in ra của chương trình trong các trường hợp sau:

(a) nếu a và i được truyền bằng **trị-kết quả**.

(b) nếu a và i được truyền bằng **tham khảo**.

(c) nếu a và i được truyền bằng **tên**.

HẾT

Scheduled Subroutine:

1) The execution of callee is NOT started when it is invoked

2) Scheduled by time or priority

3) Controlled by a scheduler

Recursive:

No explicit call site, used in
event-driven programming, error
handler

Exception:

1) No explicit call site

2) Used in Event-Driven Programming, Error Handling

Ex: By user interaction
Click, MouseMove, TextChange

By operating system
By an object (Timer)

By programmer (throw)

Coroutines:

A coroutine may postpone its execution and control is back to caller. Its execution later is resumed at the place it postponed

Tasks:

1) Able to execute concurrently with other tasks
2) Run on multi-processor machine or
single processor machine using time sharing

!!! Issue:
+ Synchronization
1) Race condition
2) Deadlock
+ Communication

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: 20-12-2018

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

Mã đề: 1182

- 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. (L0.3) Viết phương thức **visitDowhile(ast:Dowhile,o:Context)** và phương thức **visitContinue(ast:Continue,o:Context)** để sinh mã Jasmin cho một phát biểu Dowhile và Continue; và giải thích làm sao một phát biểu **continue** ở sâu bên trong thân của phát biểu **Dowhile** có thể chuyển điều khiển đến vị trí thích hợp. Nhắc lại, phát biểu do while sẽ thực thi lặp các phát biểu giữa do và while (trong body)cho đến khi điều kiện sau while (expr)trở nên sai. Cho biết lớp của phát biểu Dowhile và Continue trên AST được định nghĩa như sau:

```
class Dowhile(Stmt):
        def __init__(self,body:list(Stmt),expr: Expr)
class Continue(Stmt):
```

Một số phương thức của Emitter có thể sử dụng (nhưng không giới hạn):

- emitIFTRUE(self, label: Int, frame)
- emitIFFALSE(self, label: Int, frame)
- emitGOTO(self, label: Int, frame)
- emitLABEL(self, label: Int, frame)
- emitPUSHICONST(self, in: Int, frame)
- emitOROP(self, frame)
- emitADDOP(self, op, typ,frame)
- emitMULOP(self, op, typ,frame)

Một số phương thức của Frame có thể sử dụng(nhưng không giới hạn):

- enterLoop(self)
- exitLoop(self)
- getNewLabel(self)
- getBreakLabel(self)
- getContLabel(self)
- getStartLabel(self)
- getEndLabel(self)

2. (LO.3) Giả sử biểu thức nhị phân có các phép toán +, /, OR, trong đó, các phép toán +, / có thể có các toán hạng ở kiểu IntType hoặc FloatType, trong khi phép toán OR chỉ cho phép các toán hạng ở kiểu BoolType. Khi các toán hạng khác kiểu, phải sinh mã chuyển đổi sang kiểu FloatType. Kết quả của phép + và OR có cùng kiểu toán hạng, trong khi phép / luôn trả về kiểu FloatType. Hãy viết **visitBinaryOp(ast:BinaryOp, o:Context)** để sinh mã Jasmin cho biểu thức nhị phân. Chú ý phải sinh mã cho phép **rút ngắn tính toán (short-circuit)** cho OR thì mới được trọn điểm câu này. Nhắc lại khai báo lớp BinaryOp trên cây AST như sau:

```
class BinaryOp(Expr):
        def __init__(self,op:String,left:Expr,right: Expr)
```

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

II. Phần câu hỏi tự luận:(8 điểm)

3. (LO.2) 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);
}
```

Biểu thức điều kiện của phát biểu if phải có kiểu **boolean**, các toán hạng của phép toán == phải cùng kiểu. Kiểu của biểu thức sau **return** phải cùng kiểu trả về của hàm.

4. (LO.2) Cho đoạn mã sau được viết trên ngôn ngữ Scala dùng **qui tắc tầm vực tĩnh (static-scope rule)**. Nhắc lại, trên Scala, từ khoá **var** để khai báo biến, **def** để khai báo hàm, Int=>Int là kiểu hàm nhận vào 1 giá trị nguyên trả về 1 giá trị nguyên, giá trị của biểu thức cuối cùng trong một hàm là giá trị trả về của hàm.

```
def main = {
   var a = 0; var b = 1; var c = 5 //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(2,3,sub3)
   }
   sub1(5)
   print(b) // 5
}
```

   (a) Cho biết môi trường tham khảo tĩnh (static referencing environment) của các hàm **main, sub1, sub2, sub3** (với các tên a, b và c phải ghi kèm // dòng khai báo của tên, ví dụ a//1).

   (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? Trình bày tiếp sự thực thi cho đến khi in giá trị b của bản hoạt động main ở dòng // 5?

5. (LO.2) 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.

6. (LO.2) Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:

```
int A[3] = {5,9,41};// index of A starts from 0
int j = 0;
int n = 3;
int sumAndDecrease(int a, int i) {
     int s = 0;
     for ( ; i < n; i = i + 1) {
        s = s + a;
```

```
        A[j] = A[j] - 1;
    }
    return s;
}
void main(){
    int s = sumAndDecrease(A[j],j);
    cout << s << A[0] << A[1] << A[2]; //1
}
```

Hãy cho biết và giải thích kết quả in ra của chương trình trong các trường hợp sau:

(a) nếu a và i được truyền bằng **trị-kết quả**.

(b) nếu a và i được truyền bằng **tham khảo**.

(c) nếu a và i được truyền bằng **tên**.

7. (LO.2) Hãy trình bày về kiểu con trỏ (pointer) và kiểu tham khảo (reference)? Cho ví dụ cho mỗi kiểu? Nêu các điểm khác biệt của hai loại kiểu này? Hãy giải thích vì sao các loại kiểu này gây ra hiện tượng alias?

8. (LO.2) Hãy nêu điểm khác biệt chính của các cơ chế gọi chương trình con: đệ qui (recursive), biến cố (exception), trình cộng hành (coroutine), trình định thời (scheduled subroutine) và công tác (task) so với cơ chế Gọi-Trở về đơn giản (simple call-return)?

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

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