



KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH
BỘ MÔN KHOA HỌC MÁY TÍNH

Họ và tên:.....
MSSV:.....

Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình

Thời gian: 120 phút

Ngày thi: 23-12-2015

☐ Sinh viên được phép sử dụng tài liệu

☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1151

- Sinh viên phải ghi tên và mã số sinh viên trên đề thi (**trang đầu và trang cuối**), giấy làm bài trắc nghiệm (giấy đỏ) và giấy làm bài. Sinh viên phải tô phần mã đề thi và mã số sinh viên trên giấy làm bài trắc nghiệm. Khi nộp bài, sinh viên phải nộp cả **đề thi, giấy làm bài trắc nghiệm và giấy làm bài**.
- Phần trắc nghiệm sẽ được chấm TỰ ĐỘNG trên giấy làm bài trắc nghiệm. Do đó, phần trắc nghiệm nếu làm trên đề thi sẽ KHÔNG được chấm.
- Đối với các câu hỏi phần trắc nghiệm, sinh viên chỉ chọn MỘT phương án đúng nhất.
- Đối với phần câu hỏi tự luận (phần II), sinh viên làm ngay trên đề thi, ở phần dành riêng ngay dưới mỗi câu hỏi.
- Đối với câu hỏi bài tập lớn (phần III và IV), sinh viên làm trên giấy làm bài.
- Sinh viên lớp đại trà làm 3 phần (I, II và III). Sinh viên lớp tài năng làm cả 4 phần (I, II, III và IV). Chỉ có phần I và II được dùng để tính điểm cuối kỳ, các phần III và IV để tính điểm bài tập lớn 2 và 3.

I. Phần câu hỏi trắc nghiệm:(8 điểm))

1. Giả sử chương trình có một lỗi văn phạm nằm sau lệnh **print**. Khi em nhấn nút Run (để dịch và chạy chương trình) trên trình soạn thảo, kết quả được in ra bởi lệnh **print** trên trước khi lỗi văn phạm được báo. Hỏi chế độ dịch của trình soạn thảo là gì?

- (A) trình liên kết (link editor) (B) trình biên dịch (compiler)
(C) trình thông dịch(interpreter) (D) trình hợp ngữ (assembler)
(E) trình biên dịch động(just-in-time compiler)

2. Cho biết mã Jasmin của biểu thức viết bằng ngôn ngữ C như sau:

$a * b - 2$

với a và b là 2 biến nguyên có chỉ số lần lượt là 0 và 1.

- | | | | | |
|--------------|----------|----------|----------|----------|
| iload_0 | iload_0 | iload_0 | iload_0 | imul |
| iload_1 | iload_1 | imul | iload_1 | load_0 |
| imul | iconst_2 | iload_1 | iconst_2 | iload_1 |
| (A) iconst_2 | (B) imul | (C) isub | (D) isub | (E) isub |
| isub | isub | iconst_2 | imul | iconst_2 |
| ... | ... | ... | ... | ... |



Đoạn code sau, được viết trên ngôn ngữ dùng **qui tắc tầm vực tĩnh (static-scope rule)**, áp dụng cho các câu 3–6

```
procedure main(){
  var a, b, c:integer; //1
  procedure sub1(a: integer) { //2
    procedure sub3();
    procedure sub2() {
      var a, c : real;//3
      sub3();
    }
    procedure sub3() {
      a // use a
    }
    sub2();
  }
  sub1(3);
}
```

3. Môi trường tham khảo tĩnh (static referencing environment) của thủ tục **sub3** KHÔNG chứa

- (A) a ở //1 (B) a ở //2 (C) sub1 (D) sub2 (E) sub3

4. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong **sub3** ứng với khai báo

- (A) Báo lỗi a chưa khai báo (B) a trong sub1 (//2) (C) a trong sub2 (//3)
(D) a trong sub3 (E) a trong main (//1)

5. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong sub3 **khi thực thi** ứng với kết hợp của **a** trong bản hoạt động của

- (A) main (B) sub1 (C) sub2 (D) sub3
(E) Báo lỗi không tìm thấy a

6. Nếu đoạn code trên được viết trên ngôn ngữ dùng qui tắc tầm vực động (dynamic-scope rule), và giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến a trong sub3 **khi thực thi** ứng với kết hợp của a trong bản hoạt động của

- (A) Báo lỗi không tìm thấy a (B) main (C) sub1 (D) sub2
(E) sub3

Cho văn phạm của một biểu thức sau dùng cho các câu 7–9:

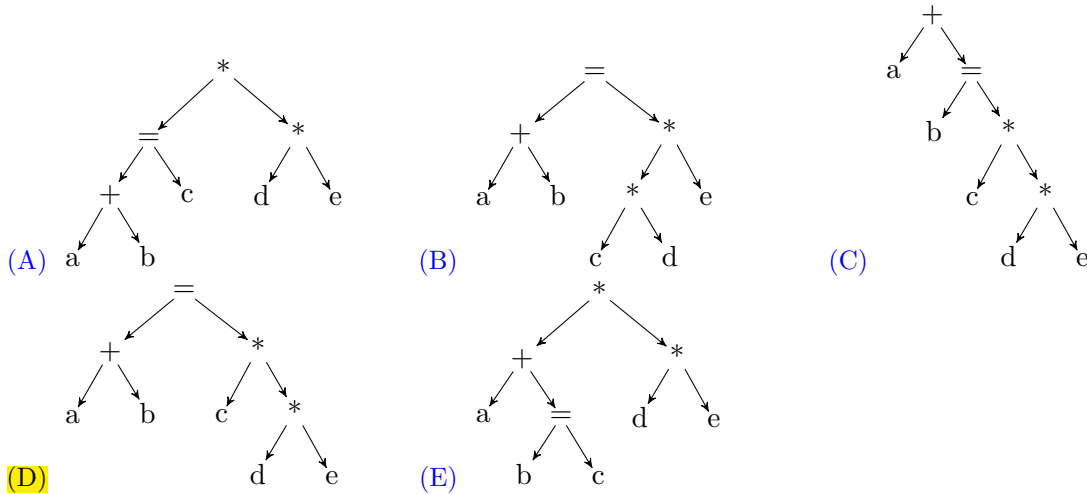
```
exp  → term '=' exp | term
term → term '+' fact | term '>' fact | fact
fact → ope '*' ope | ope
ope  → '(' exp ')' | ID
```

với ID là một danh hiệu.

7. Chọn biểu thức được viết đúng văn phạm, có số lượng dấu '(' và ')' ít nhất mà tương đương với biểu thức sau: $(a + (b = ((c * d) * e))) > f$

- (A) $(a + (b = (c * d) * e)) > f$ (B) $(a + (b = ((c * d) * e))) > f$
(C) $a + b = c * d * e > f$ (D) $a + (b = (c * d) * e) > f$
(E) $a + (b = c * d * e) > f$

8. Vẽ AST cho biểu thức sau được viết dựa vào văn phạm trên: $a + b = c * (d * e)$



9. Cho cấu trúc AST được khai báo trên Scala như sau:

```
trait Exp
case class Bin(op:String,e1:Exp,e2:Exp) extends Exp
case class Id(i:String) extends Exp
```

Chọn AST cho biểu thức sau: $a + b > c = d * e = f$

- (A) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Bin("=",Id("e"),Id("f"))))`
- (B) `Bin("=",Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Id("e"))),Id("f"))`
- (C) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("=",Bin("*",Id("d"),Id("e")),Id("f"))`
- (D) `Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("=",Bin("*",Id("d"),Id("e")),Id("f"))`
- (E) `Bin("=",Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("*",Id("d"),Id("e"))),Id("f"))`

Phần trình bày sau dùng trong các câu hỏi 10– 11:

Cho mã giả của phát biểu `for var = expr1 to expr2 do body` như sau:

```
start:  var = expr1
        etemp = expr2
loop:   if (var >= etemp) goto out
        body
label1: var++
label2: goto loop
out:
```

10. Nếu trong thân *body* có thực thi lệnh *continue* thì điều khiển sẽ chuyển đến thực thi phát biểu có nhãn là:

- (A) out (B) start (C) loop (D) label1 (E) label2

11. Với hiện thực của phát biểu *for* như trên, phát biểu nào đặt trong thân phát biểu **for** dưới đây sẽ làm cho phát biểu này bị lặp mãi mãi?

```
n = 10; s = 0;
for i = s to n do -----
```

- (A) `i = i - 1;` (B) `s = s - 1;` (C) `n = n + 1;`
- (D) câu A và C đúng
- (E) Không phát biểu nào trong các câu A, B và C làm phát biểu **for** lặp mãi mãi

12. Cho biết mã Jasmin của phát biểu gán sau:

$a[i] = a[i] + 2$

với i kiểu nguyên có chỉ số là 3 và a là kiểu dãy nguyên có chỉ số là 1.

			aload_1		aload_1		aload_1
	iastore		iload_3		dup		iload_3
	iadd		aload_1		iload_3		iastore
	iaload		iload_3		dup		aload_1
(A)	aload_1	(B)	iconst_2	(C)	iaload	(D)	iload_3
	iload_3		iadd		iconst_2		iconst_2
	dup		iastore		iadd		iadd
	iconst_2		...		iastore		...

Đoạn code sau dùng cho các câu 13–15

```
int p;
int* foo(int x) {
    static int q;
    int *s = new int;
    switch (x) {
        case 1: return &p;
        case 2: return &q;
        case 3: return &x;
        case 4: return s;
        default: return foo(x-1);
    }
}
```

13. Phát biểu nào sẽ gây ra lỗi tham chiếu treo (dangling reference) khi thực thi

- (A) return &p (B) return &q (C) return &x (D) return s
(E) Không phát biểu nào gây ra lỗi trên khi thực thi

14. Trong đoạn mã trên, đối tượng dữ liệu nào có thể trở thành rác (garbage)

- (A) p (B) q (C) x (D) trở đến bởi s
(E) Không có đối tượng nào có thể trở thành rác

15. Khi hàm **foo** được gọi đệ quy (recursive) thì các bản hoạt động của **foo** dùng chung những đối tượng dữ liệu nào?

- (A) q và s (B) p, q, x và s (C) p (D) p và q (E) x, q và s

16. Viết lại biểu thức trung tố (infix) sau sang dạng hậu tố (postfix) Polish? Giả sử là tất cả các phép toán đều có 2 toán hạng và có độ ưu tiên và tính kết hợp như trong ngôn ngữ lập trình C.

$a + b * c - d * e + f$

- (A) $a + b * c * d * e * - f +$ (B) $a b + c * d - e * f +$ (C) $a b c * + d - e f * +$
(D) $a b c * + d e * + f -$ (E) $a b c * + d e * - f +$

17. Cho khai báo sau được viết trên ngôn ngữ lập trình Ada:

```

type Shape is ( Circle , Triangle , Rectangle );
type Colors is ( Red, Green , Blue );
type Figure ( Form: Shape ) is record
    Filled: Boolean;
    Color: Colors;
    case Form is
        when Circle => Diameter: Float;
        when Triangle =>
            Leftside , Rightside: Integer;
            Angle: Float;
        when Rectangle => Side1 , Side2: Integer;
    end case;
end record;

```

Cho kích thước của các kiểu **Integer, Float, Boolean và Enumeration** lần lượt là 2, 4, 1, và 2, và kích thước cho phần mô tả kiểu (type description) là 0. Cho biết kích thước của một đối tượng kiểu **Figure**?

- (A) 21 (B) 13 (C) 23 (D) 15 (E) Khác

18. Cơ chế gọi chương trình nào là cơ chế gọi chương trình con cơ bản của Lập trình hướng sự kiện (Event-driven Programming)

- (A) Biến cố - Xử lý biến cố (Exception)
 (B) Gọi trở về đơn giản (Simple Call - Return)
 (C) Gọi đệ qui (Recursive call)
 (D) Trình cộng hành (Coroutine)
 (E) Trình định thời (Scheduled Subprogram)

19. Đoạn code viết bằng Scala nào dưới đây có thể dùng như một ví dụ minh họa cho khái niệm Hàm Currying

- (A) `def add1(n:Int) = n + 1`
 (B) `List(1,2,3).foldLeft(0)(_ + _)`
 (C) `def add(n:Int) = (x:Int) => n + x; val add2 = add(2)`
 (D) `def add(n:Int)(x:Int) = n + x; val add2 = add (2) _`
 (E) `List(1,2,3).filter(_ > 1)`

20. Liệt kê theo thứ tự chiều dài tăng dần ít nhất 5 chuỗi ngắn nhất (nếu có nhiều hơn 5 chuỗi) của ngôn ngữ được mô tả bởi biểu thức chính quy (regular expression) sau:
 $ab^*(a|b)?a$

- (A) aa, aba, aaa, abaa, aaba (B) ϵ , a, aa, aba, abaa (C) aa, aba, aaa, abaa, abba
 (D) a, aa, aba, aaa, aaba (E) aa, aba, aaa, abaa, aaaa

21. Cho giá trị ban đầu của biến **c** là 3, cho biết những giá trị **có thể có** của biến **a** sau khi thực hiện phép gán sau trên ngôn ngữ lập trình C?

`a = c * (c = 5);`

- (A) 15, 25, 9 (B) 15 (C) 25 (D) 9 (E) 15, 25

22. Cho biết kích thước của đối tượng dữ liệu **x** được khai báo như sau:

`x: set of 1..16`

- (A) 4 bytes (B) 2 bytes (C) 4 bits (D) 1 byte (E) 16 bytes

23. Cho lớp(class) A là lớp cha (superclass) của lớp B và B là lớp cha của hai lớp C và D. Trong lớp A, có khai báo phương thức thực thể (instance method) **foo** và phương thức này bị ghi đè(override) trên tất cả các lớp B, C và D. Cho biến x được khai báo như sau:
B x;

Qui ước viết $func_T$ là phương thức func được khai báo trong lớp T, cho biết tập của phương thức đích của cuộc gọi sau: **x.foo();**

- (A) $\{foo_C, foo_D\}$ (B) $\{foo_A, foo_B, foo_C, foo_D\}$ (C) $\{foo_B\}$
(D) $\{foo_A, foo_B\}$ (E) $\{foo_B, foo_C, foo_D\}$

24. Chọn phát biểu ĐÚNG trong các phát biểu về dãy (array) sau:

- (A) Trên C, kích thước của dãy có thể tăng thêm sau khi dãy đã được tạo ra
(B) Truy xuất một phần tử bên ngoài một dãy C không được phép, sẽ lập tức gây ra lỗi
(C) Trên C và C++, kích thước của một dãy được lưu trữ để kiểm tra động
(D) Trên Java, kích thước của một dãy KHÔNG được lưu trữ để kiểm tra động
(E) Truy xuất một phần tử bên ngoài một dãy Java không được phép, sẽ lập tức gây ra lỗi

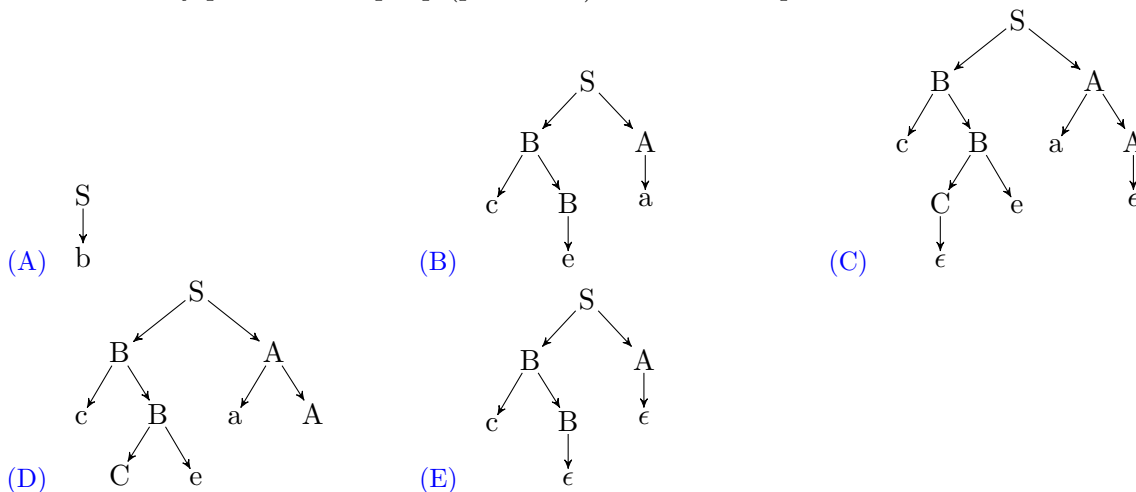
Cho văn phạm sau dùng cho các câu 25–26:

S \rightarrow b | B A
A \rightarrow a A | ϵ
B \rightarrow C e | c B
C \rightarrow d C | ϵ

25. Chuỗi nào KHÔNG thuộc ngôn ngữ được mô tả bởi văn phạm trên?

- (A) e (B) b (C) eaaa (D) cccddddeaa (E) cccaaa

26. Chọn cây phân tích cú pháp (parse tree) cho chuỗi nhập: **cea**



Phần hướng dẫn này áp dụng cho các câu 27–31

Hãy viết thêm các đoạn mã cần thiết vào phương thức dưới đây để thực hiện sinh mã cho phát biểu *repeat*. Nhắc lại phát biểu *repeat* kết thúc quá trình lặp khi điều kiện của phát biểu trở thành **true**. AST của phát biểu **repeat** được định nghĩa như sau:

```
case class Repeat(val stmts: List[Stmt], val exp: Expr) extends Stmt
```

```
override def visitRepeat(ast: Repeat, o: Context) = {  
    val ctxt = o.asInstanceOf[SubBody]  
    ctxt.frame.enterLoop();  
}
```

```
val labelStart = ctxt.frame.getNewLabel()
val labelBreak = ctxt.frame.getBreakLabel()
val labelCont = ctxt.frame.getContinueLabel()
val str1 = //1
// sinh mã cho từng phát biểu trong thân của repeat
val str2 = //2
val str3 = //3
// sinh mã cho biểu thức điều kiện
val str4 = visit(ast.exp,...)
val str5 = //4
val str6 = //5
frame().exitLoop();
str1 + str2 + str3 + str4 + str5 + str6
}
```

27. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //1 trong đoạn mã trên

- | | |
|-------------------------------------|-------------------------------------|
| (A) "" | (B) ctxt.emit.emitLABEL(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitLABEL(labelCont) |
| (E) ctxt.emit.emitGOTO(labelStart) | |

28. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //2 trong đoạn mã trên

- (A) visit(ast.stmts,o)
(B) ast.stmts.map(x=>visit(x,o))
(C) ast.stmts.filter(x=>visit(x,o))
(D) ast.stmts.foldLeft(")((x,y)=>x + visit(y,o))"
(E) ast.stmts.foldLeft(")((x,y)=>y + visit(x,o))"

29. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //3 trong đoạn mã trên

- | | |
|-------------------------------------|---------------------------------------|
| (A) "" | (B) ctxt.emit.emitIFFALSE(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitLABEL(labelCont) |
| (E) ctxt.emit.emitGOTO(labelStart) | |

30. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //4 trong đoạn mã trên

- | | |
|-------------------------------------|---------------------------------------|
| (A) ctxt.emit.emitGOTO(labelBreak) | (B) ctxt.emit.emitIFFALSE(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitLABEL(labelCont) |
| (E) ctxt.emit.emitGOTO(labelStart) | |

31. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //5 trong đoạn mã trên

- | | |
|-------------------------------------|--|
| (A) ctxt.emit.emitGOTO(labelBreak) | (B) ctxt.emit.emitIFFALSE(labelStart)) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitLABEL(labelCont)) |
| (E) ctxt.emit.emitGOTO(labelStart)) | |

32. Trình định thời (scheduled subprograms) thường được dùng trong

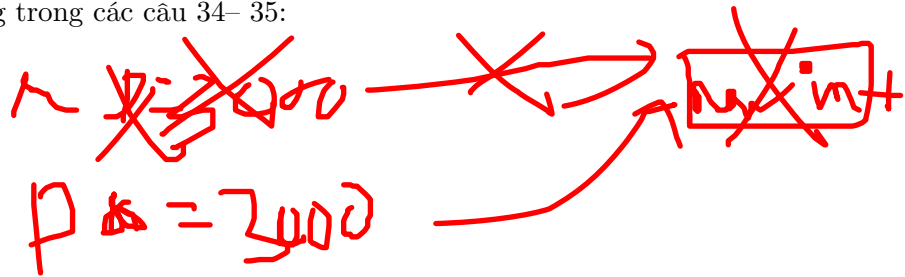
- (A) Lập trình song song (Parallel Programming)
- (B) Lập trình hướng đối tượng (Object-Oriented Programming)
- (C) Lập trình hàm (Functional Programming)
- (D) Lập trình hướng sự kiện (Event-driven Programming)
- (E) Lập trình thời gian thực (Real-time Programming)

33. Gọi append là hàm dùng để nối 2 danh sách thành 1 danh sách. Ví dụ `append([1,2,3],[4,5,6])` sẽ có kết quả là `[1,2,3,4,5,6]`. Hãy hiện thực hàm `append(a:List[Int],b:List[Int])` dùng hàm bậc cao (high-order function)?

- (A) `a.map(b)((x,y)=>x::y)`
- (B) `a.foldLeft(b)((x,y)=>x::y)`
- (C) `a.foldRight(b)((x,y)=>x::y)`
- (D) `b.foldLeft(a)((x,y)=>x::y)`
- (E) `b.foldRight(a)((x,y)=>x::y)`

Đoạn mã sau được dùng trong các câu 34– 35:

```
int *p = new int;
void foo(int * r) {
    delete r;
}
foo(p); //1
*p = 2; //2
```



34. Hiện tượng gì xảy ra khi `p` được truyền cho `r` ở phát biểu `//1` trong đoạn mã trên:

- (A) Con trỏ chưa khai báo (undeclared pointer)
- (B) Bí danh (alias)
- (C) Khai báo trùng tên (redeclared)
- (D) Tham chiếu treo (dangling reference)
- (E) Rác (garbage)

35. Hiện tượng gì xảy ra khi thực thi phát biểu `//2` trong đoạn mã trên:

- (A) Tham chiếu treo
- (B) Bí danh
- (C) Đa hình (polymorphism)
- (D) Rác
- (E) Con trỏ chưa khai báo

Đoạn mã sau được dùng cho các câu 36–41.

Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:

```
int A[5] = {1,3,5,7,9}; // index of A starts from 0
int j = 0;
int n = 5;
int sumAndIncrease(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 = sumAndIncrease(A[j], j);
    printf("a = %i\n", s); //1
    printf(" %i %i %i %i %i\n", A[0], A[1], A[2], A[3], A[4]); //2
}
```




36. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 (B) 10 (C) 15 (D) 25 (E) Khác

37. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 2 4 6 8 10 (C) 6 3 5 7 9 (D) 5 3 5 7 9 (E) Khác

38. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 (B) 10 (C) 15 (D) 25 (E) Khác

39. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 2 4 6 8 10 (C) 6 3 5 7 9 (D) 5 3 5 7 9 (E) Khác

40. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 (B) 15 (C) 20 (D) 25 (E) Khác

41. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 2 4 6 8 10 (C) 6 3 5 7 9 (D) 5 3 5 7 9 (E) Khác

42. Một biến trên ngôn ngữ Javascript có thể nhận những giá trị thuộc các kiểu khác nhau. Ví dụ:

```
var x = "def";
```

```
x = 1;
```

Ràng buộc giữa một biến và kiểu của nó trên ngôn ngữ Javascript xảy ra trong thời gian

- (A) Hiện thực (implementation) (B) Chạy (running)
(C) Lập trình (programming) (D) Dịch (compiling)
(E) Khởi động (loading)

43. Cho $X = \{a, b\}$. Chọn biểu thức chính quy (regular expression) mô tả ngôn ngữ chứa bất kỳ chuỗi nào được tạo bởi các ký tự trên tập X nhưng KHÔNG chứa chuỗi có 2 ký tự a liên tiếp.

- (A) $a|(abb^*)^*$ (B) $(b^*ab^*)^*$ (C) $b^*(abb^*)^*$ (D) $b^*a?(bb^*a)^*b^*$ (E) $b^*ab^*ab^*$

44. Kiểu *union* của ngôn ngữ C++ cho phép ghi vào ở thành phần này nhưng đọc lên ở thành phần khác như trong ví dụ sau:

```
union {  
    int songuyen;  
    char kytu[2];  
} x;  
x.songuyen = 12;  
cout << x.kytu[0] << " " << x.kytu[1] << endl;
```

Đặc tính này của ngôn ngữ C++ gây ảnh hưởng XẤU NHẤT đối với

- (A) tính trực giao (orthogonality) (B) tính đơn giản (simplicity)
(C) tính dễ viết (writability) (D) tính tin cậy (reliability)
(E) chi phí (cost)

45. Cơ chế gọi chương trình con nào mà điều khiển có thể chuyển vào một vị trí ở giữa chương trình con được gọi thay vì chuyển vào đầu chương trình con được gọi?

- (A) Trình định thời (Scheduled Subprogram)
- (B) Gọi trở về đơn giản (Simple Call - Return)
- (C) Gọi đệ qui (Recursive call)
- (D) Biến cố - Xử lý biến cố (Exception)
- (E) Trình cộng hành (Coroutine)

46. Để thực hiện tìm kiếm một chuỗi (String) trên nhiều loại danh sách khác nhau, hãy chọn giải pháp thích hợp nhất để điền vào chỗ trống trong định nghĩa hàm sau:

```
def lookup[T](x:String, lst:List[T], f:T=>String):Option[T] =  
  lst match {  
    case List() => None  
    case h::t => _____  
  }
```

- (A) if (h == x) Some(h) else lookup(x,t,f)
- (B) if (f(h) == x) Some(h) else lookup(x,t,f)
- (C) if (h == f(x)) Some(h) else lookup(x,t,f)
- (D) if (f(h) == x) Some(f(h)) else lookup(x,t,f)
- (E) if (h == f(x)) Some(f(h)) else lookup(x,t,f)

47. Khi cần khai thác đặc tính của kiểu union (các thành phần loại trừ lẫn nhau), làm thế nào để hiện thực trên những ngôn ngữ lập trình hướng đối tượng như Java (không có kiểu union)? Cụ thể làm thế nào để hiện thực biến z kiểu union sau trên những ngôn ngữ này?

```
union {  
  string loikhen;  
  string loiche;  
} z;
```

- (A) Định nghĩa lớp cha A và hai lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A
- (B) Định nghĩa một lớp A có 2 thuộc tính loikhen và loiche; và z có kiểu A
- (C) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu A
- (D) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu B
- (E) Định nghĩa lớp trừu tượng (abstract class) A và 2 lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A

II. Phần câu hỏi tự luận: (dành cho tất cả sinh viên) (2 điểm)

48. Trên những ngôn ngữ lập trình cho phép rút ngắn tính toán (short-circuit evaluation) biểu thức luận lý, biểu thức sau sẽ không gây ra lỗi khi thực thi.

((a != 0) && ((b / a) > 5))

Hãy dùng **if then else** để mô phỏng quá trình tính toán của biểu thức luận lý trên?



49. Hãy nêu những giải pháp đã được sử dụng để tránh tham chiếu treo (dangling reference) trong trường hợp trả về hàm như sau:

```
void->int F() {  
    int x = 1;  
    int g() {  
        return x + 1;  
    }  
    return g ;  
}  
void->int gg = F();  
int z = gg();
```

50. 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



III. Phần bài tập lớn: (dành cho tất cả sinh viên)

51. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện kiểm tra kiểu. Hàm `visitBlock` sẽ thực hiện chức năng xây dựng bảng danh hiệu (symbol table) và truyền bảng này khi visit các phát biểu trong khối. Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm. Nhắc lại AST của Block được khai báo như sau:

```
case class Block(val decls:List[Decl],val stmts:List[Stmt]) extends Stmt
```

52. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện sinh mã cho một phát biểu If. Nhắc lại AST của If được khai báo như sau:

```
case class If(val expr:Expr, val thenStmt:Stmt, val elseStmt:Option[Stmt]) extends Stmt
```

Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm.

IV. Phần KSTN: (dành cho sinh viên KSTN)

53. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện kiểm tra lỗi FunctionNotReTurn? Yêu cầu giải thích ý tưởng thực hiện thể hiện qua mã của hàm?

54. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện sinh mã cho một khối (block)? Yêu cầu giải thích ý tưởng thực hiện sinh mã cho các khai báo array trong một khối?

HẾT

Họ và tên:.....	Điểm:
MSSV:.....	



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



KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH
BỘ MÔN KHOA HỌC MÁY TÍNH

ĐÁP ÁN cho Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình

Thời gian: 120 phút

☐ Sinh viên được phép sử dụng tài liệu

☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1151

I. Phần câu hỏi trắc nghiệm:(8 điểm))

1. Giả sử chương trình có một lỗi văn phạm nằm sau lệnh **print**. Khi em nhấn nút Run (để dịch và chạy chương trình) trên trình soạn thảo, kết quả được in ra bởi lệnh **print** trên trước khi lỗi văn phạm được báo. Hỏi chế độ dịch của trình soạn thảo là gì?

- (A) trình liên kết (link editor) (B) trình biên dịch (compiler)
(C) trình thông dịch(interpreter) (D) trình hợp ngữ (assembler)
(E) trình biên dịch động(just-in-time compiler)

2. Cho biết mã Jasmin của biểu thức viết bằng ngôn ngữ C như sau:

$a * b - 2$

với a và b là 2 biến nguyên có chỉ số lần lượt là 0 và 1.

- | | | | | |
|-------------|--------------|-------------|--------------|-------------|
| (A) iload_0 | iload_0 | iload_0 | iload_0 | imul |
| iload_1 | iload_1 | imul | iload_1 | load_0 |
| imul | (B) iconst_2 | (C) iload_1 | (D) iconst_2 | (E) iload_1 |
| iconst_2 | imul | isub | isub | isub |
| isub | isub | iconst_2 | imul | iconst_2 |
| ... | ... | ... | ... | ... |

Đoạn code sau, được viết trên ngôn ngữ dùng **qui tắc tầm vực tĩnh (static-scope rule)**, áp dụng cho các câu 3–6

```
procedure main(){
  var a, b, c:integer; //1
  procedure sub1(a: integer) { //2
    procedure sub3();
    procedure sub2() {
      var a, c : real; //3
      sub3();
    }
    procedure sub3() {
      a // use a
    }
    sub2();
  }
  sub1(3);
}
```

3. Môi trường tham khảo tĩnh (static referencing environment) của thủ tục **sub3** KHÔNG chứa

- (A) a ở //1 (B) a ở //2 (C) sub1 (D) sub2 (E) sub3

4. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong **sub3** ứng với khai báo

- (A) Báo lỗi a chưa khai báo (B) a trong sub1 (//2) (C) a trong sub2 (//3)
(D) a trong sub3 (E) a trong main (//1)

5. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong sub3 **khi thực thi** ứng với kết hợp của **a** trong bản hoạt động của

- (A) main (B) sub1 (C) sub2 (D) sub3
(E) Báo lỗi không tìm thấy a

6. Nếu đoạn code trên được viết trên ngôn ngữ dùng qui tắc tầm vực động (dynamic-scope rule), và giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến a trong sub3 **khi thực thi** ứng với kết hợp của a trong bản hoạt động của

- (A) Báo lỗi không tìm thấy a (B) main (C) sub1 (D) sub2
(E) sub3

Cho văn phạm của một biểu thức sau dùng cho các câu 7-9:

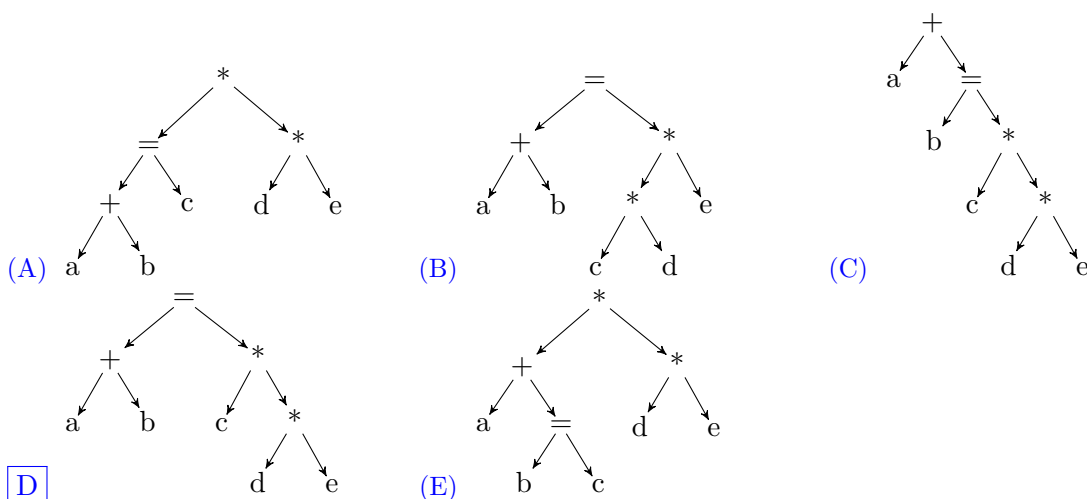
exp \rightarrow term '=' exp | term
term \rightarrow term '+' fact | term '>' fact | fact
fact \rightarrow ope '*' ope | ope
ope \rightarrow '(' exp ')' | ID

với ID là một danh hiệu.

7. Chọn biểu thức được viết đúng văn phạm, có số lượng dấu '(' và ')' ít nhất mà tương đương với biểu thức sau: $(a + (b = ((c * d) * e))) > f$

- (A) $(a + (b = (c * d) * e)) > f$ (B) $(a + (b = ((c * d) * e))) > f$
(C) $a + b = c * d * e > f$ (D) $a + (b = (c * d) * e) > f$
(E) $a + (b = c * d * e) > f$

8. Vẽ AST cho biểu thức sau được viết dựa vào văn phạm trên: $a + b = c * (d * e)$





9. Cho cấu trúc AST được khai báo trên Scala như sau:

```
trait Exp
case class Bin(op:String,e1:Exp,e2:Exp) extends Exp
case class Id(i:String) extends Exp
```

Chọn AST cho biểu thức sau: $a + b > c = d * e = f$

- (A) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("=",Bin("*",Id("d"),Bin("=",Id("e"),Id("f")))))`
 (B) `Bin("=",Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Id("e"))),Id("f"))`
 (C) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("=",Bin("*",Id("d"),Id("e"))),Id("f"))`
 (D) `Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("=",Bin("*",Id("d"),Id("e"))),Id("f"))`
 (E) `Bin("=",Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("*",Id("d"),Id("e"))),Id("f"))`

Phần trình bày sau dùng trong các câu hỏi 10– 11:

Cho mã giả của phát biểu *for var = expr1 to expr2 do body* như sau:

```
start:  var = expr1
        etemp = expr2
loop:   if (var >= etemp) goto out
        body
label1: var++
label2: goto loop
out:
```

10. Nếu trong thân *body* có thực thi lệnh *continue* thì điều khiển sẽ chuyển đến thực thi phát biểu có nhãn là:

- (A) out (B) start (C) loop (D) label1 (E) label2

11. Với hiện thực của phát biểu *for* như trên, phát biểu nào đặt trong thân phát biểu **for** dưới đây sẽ làm cho phát biểu này bị lặp mãi mãi?

```
n = 10; s = 0;
for i = s to n do -----
```

- (A) `i = i - 1;` (B) `s = s - 1;` (C) `n = n + 1;`
 (D) câu A và C đúng
 (E) Không phát biểu nào trong các câu A, B và C làm phát biểu **for** lặp mãi mãi

12. Cho biết mã Jasmin của phát biểu gán sau:

`a[i] = a[i] + 2`

với *i* kiểu nguyên có chỉ số là 3 và *a* là kiểu dãy nguyên có chỉ số là 1.

- | | | | | | | | | | |
|-----|---|-----|---|-----|---|-----|---|-----|--|
| (A) | <code>iastore</code>
<code>iadd</code>
<code>iaload</code>
<code>aload_1</code>
<code>iload_3</code>
<code>dup</code>
<code>iconst_2</code>
<code>...</code> | (B) | <code>aload_1</code>
<code>iload_3</code>
<code>iaload</code>
<code>iconst_2</code>
<code>iadd</code>
<code>iastore</code>
<code>...</code> | (C) | <code>aload_1</code>
<code>iload_3</code>
<code>aload_1</code>
<code>iload_3</code>
<code>iaload</code>
<code>iconst_2</code>
<code>iadd</code>
<code>iastore</code>
<code>...</code> | (D) | <code>aload_1</code>
<code>dup</code>
<code>iload_3</code>
<code>dup</code>
<code>iaload</code>
<code>iconst_2</code>
<code>iadd</code>
<code>iastore</code>
<code>...</code> | (E) | <code>aload_1</code>
<code>iload_3</code>
<code>iastore</code>
<code>aload_1</code>
<code>iload_3</code>
<code>iconst_2</code>
<code>iadd</code>
<code>...</code> |
|-----|---|-----|---|-----|---|-----|---|-----|--|



Đoạn code sau dùng cho các câu 13–15

```
int p;  
int* foo(int x) {  
    static int q;  
    int *s = new int;  
    switch (x) {  
        case 1: return &p;  
        case 2: return &q;  
        case 3: return &x;  
        case 4: return s;  
        default: return foo(x-1);  
    }  
}
```

13. Phát biểu nào sẽ gây ra lỗi tham chiếu treo (dangling reference) khi thực thi

- (A) return &p (B) return &q ☒ (C) return &x (D) return s
(E) Không phát biểu nào gây ra lỗi trên khi thực thi

14. Trong đoạn mã trên, đối tượng dữ liệu nào có thể trở thành rác (garbage)

- (A) p (B) q (C) x
☒ (D) trở đến bởi s (E) Không có đối tượng nào có thể trở thành rác

15. Khi hàm **foo** được gọi đệ quy (recursive) thì các bản hoạt động của **foo** dùng chung những đối tượng dữ liệu nào?

- (A) q và s (B) p, q, x và s (C) p ☒ (D) p và q
(E) x, q và s

16. Viết lại biểu thức trung tố (infix) sau sang dạng hậu tố (postfix) Polish? Giả sử là tất cả các phép toán đều có 2 toán hạng và có độ ưu tiên và tính kết hợp như trong ngôn ngữ lập trình C.

$a + b * c - d * e + f$

- (A) $a + b c * d e * - f +$ (B) $a b + c * d - e * f +$ (C) $a b c * + d - e f * +$
(D) $a b c * + d e * + f -$ ☒ (E) $a b c * + d e * - f +$

17. Cho khai báo sau được viết trên ngôn ngữ lập trình Ada:

```
type Shape is ( Circle , Triangle , Rectangle );
type Colors is ( Red, Green, Blue );
type Figure (Form: Shape) is record
    Filled: Boolean;
    Color: Colors;
    case Form is
        when Circle => Diameter: Float;
        when Triangle =>
            Leftside, Rightside: Integer;
            Angle: Float;
        when Rectangle => Side1, Side2: Integer;
    end case;
end record;
```

Cho kích thước của các kiểu *Integer*, *Float*, *Boolean* và *Enumeration* lần lượt là 2, 4, 1, và 2, và kích thước cho phần mô tả kiểu (type description) là 0. Cho biết kích thước của một đối tượng kiểu **Figure**?

- (A) 21 ☒ (B) 13 (C) 23 (D) 15 (E) Khác

18. Cơ chế gọi chương trình nào là cơ chế gọi chương trình con cơ bản của Lập trình hướng sự kiện (Event-driven Programming)

- ☒ (A) Biến cố - Xử lý biến cố (Exception)
(B) Gọi trở về đơn giản (Simple Call - Return)
(C) Gọi đệ qui (Recursive call)
(D) Trình cộng hành (Coroutine)
(E) Trình định thời (Scheduled Subprogram)

19. Đoạn code viết bằng Scala nào dưới đây có thể dùng như một ví dụ minh họa cho khái niệm Hàm Currying

- (A) `def add1(n:Int) = n + 1`
(B) `List(1,2,3).foldLeft(0)(_ + _)`
(C) `def add(n:Int) = (x:Int) => n + x; val add2 = add(2)`
☒ (D) `def add(n:Int)(x:Int) = n + x; val add2 = add (2) _`
(E) `List(1,2,3).filter(_ > 1)`

20. Liệt kê theo thứ tự chiều dài tăng dần ít nhất 5 chuỗi ngắn nhất (nếu có nhiều hơn 5 chuỗi) của ngôn ngữ được mô tả bởi biểu thức chính quy (regular expression) sau:
 $ab^*(a|b)?a$

- (A) aa, aba, aaa, abaa, aaba (B) ϵ , a, aa, aba, abaa ☒ (C) aa, aba, aaa, abaa, abba
(D) a, aa, aba, aaa, aaba (E) aa, aba, aaa, abaa, aaaa

21. Cho giá trị ban đầu của biến **c** là 3, cho biết những giá trị **có thể có** của biến **a** sau khi thực hiện phép gán sau trên ngôn ngữ lập trình C?
`a = c * (c = 5);`

- (A) 15, 25, 9 (B) 15 (C) 25 (D) 9 ☒ (E) 15, 25



22. Cho biết kích thước của đối tượng dữ liệu **x** được khai báo như sau:
x: set of 1..16

- (A) 4 bytes ☒ (B) 2 bytes (C) 4 bits (D) 1 byte
(E) 16 bytes

23. Cho lớp(class) A là lớp cha (superclass) của lớp B và B là lớp cha của hai lớp C và D. Trong lớp A, có khai báo phương thức thực thể (instance method) **foo** và phương thức này bị ghi đè(override) trên tất cả các lớp B, C và D. Cho biến x được khai báo như sau:
B x;

Qui ước viết $func_T$ là phương thức func được khai báo trong lớp T, cho biết tập của phương thức đích của cuộc gọi sau: **x.foo();**

- (A) $\{foo_C, foo_D\}$ (B) $\{foo_A, foo_B, foo_C, foo_D\}$ (C) $\{foo_B\}$
(D) $\{foo_A, foo_B\}$ ☒ (E) $\{foo_B, foo_C, foo_D\}$

24. Chọn phát biểu ĐÚNG trong các phát biểu về dãy (array) sau:

- (A) Trên C, kích thước của dãy có thể tăng thêm sau khi dãy đã được tạo ra
(B) Truy xuất một phần tử bên ngoài một dãy C không được phép, sẽ lập tức gây ra lỗi
(C) Trên C và C++, kích thước của một dãy được lưu trữ để kiểm tra động
(D) Trên Java, kích thước của một dãy KHÔNG được lưu trữ để kiểm tra động
☒ (E) Truy xuất một phần tử bên ngoài một dãy Java không được phép, sẽ lập tức gây ra lỗi

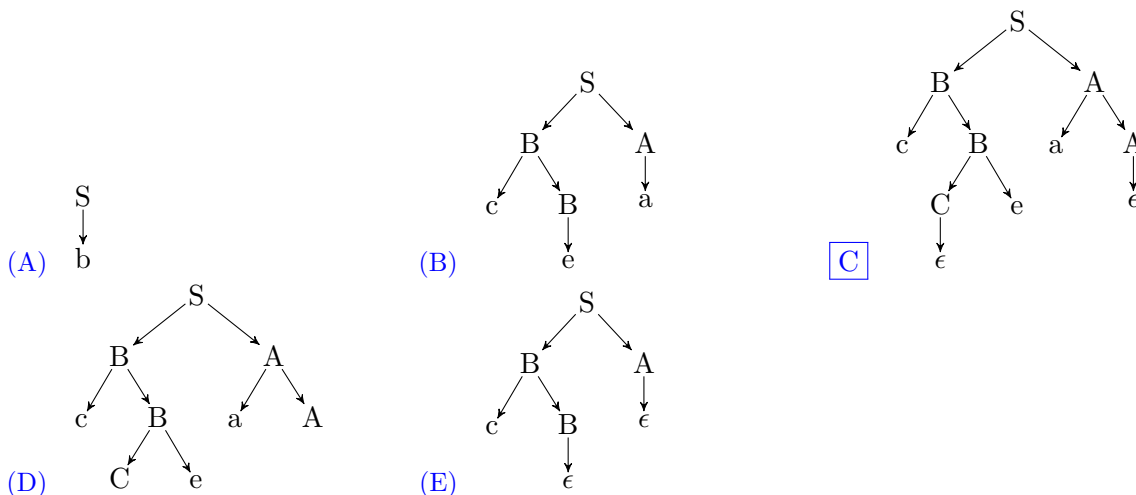
Cho văn phạm sau dùng cho các câu 25–26:

S \rightarrow b | B A
A \rightarrow a A | ϵ
B \rightarrow C e | c B
C \rightarrow d C | ϵ

25. Chuỗi nào KHÔNG thuộc ngôn ngữ được mô tả bởi văn phạm trên?

- (A) e (B) b (C) eaaa (D) cccddddeaa ☒ (E) cccaaa

26. Chọn cây phân tích cú pháp (parse tree) cho chuỗi nhập: **cea**



Phần hướng dẫn này áp dụng cho các câu 27–31

Hãy viết thêm các đoạn mã cần thiết vào phương thức dưới đây để thực hiện sinh mã cho phát biểu *repeat*. Nhắc lại phát biểu *repeat* kết thúc quá trình lặp khi điều kiện của phát biểu trở thành **true**. AST của phát biểu **repeat** được định nghĩa như sau:

```
case class Repeat(val stmts: List[Stmt], val exp: Expr) extends Stmt
```

```
override def visitRepeat(ast: Repeat, o: Context) = {  
    val ctxt = o.asInstanceOf[SubBody]  
    ctxt.frame.enterLoop();  
    val labelStart = ctxt.frame.getNewLabel()  
    val labelBreak = ctxt.frame.getBreakLabel()  
    val labelCont = ctxt.frame.getContinueLabel()  
    val str1 = //1  
    // sinh mã cho từng phát biểu trong thân của repeat  
    val str2 = //2  
    val str3 = //3  
    // sinh mã cho biểu thức điều kiện  
    val str4 = visit(ast.exp, ....)  
    val str5 = //4  
    val str6 = //5  
    frame().exitLoop();  
    str1 + str2 + str3 + str4 + str5 + str6  
}
```

27. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //1 trong đoạn mã trên

- | | |
|-------------------------------------|-------------------------------------|
| (A) "" | (B) ctxt.emit.emitLABEL(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitLABEL(labelCont) |
| (E) ctxt.emit.emitGOTO(labelStart) | |

28. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //2 trong đoạn mã trên

- (A) visit(ast.stmts, o)
(B) ast.stmts.map(x => visit(x, o))
(C) ast.stmts.filter(x => visit(x, o))
(D) ast.stmts.foldLeft("")((x, y) => x + visit(y, o))
(E) ast.stmts.foldLeft("")((x, y) => y + visit(x, o))

29. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //3 trong đoạn mã trên

- | | |
|-------------------------------------|---------------------------------------|
| (A) "" | (B) ctxt.emit.emitIFFALSE(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitLABEL(labelCont) |
| (E) ctxt.emit.emitGOTO(labelStart) | |

30. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //4 trong đoạn mã trên

- | | |
|-------------------------------------|---------------------------------------|
| (A) ctxt.emit.emitGOTO(labelBreak) | (B) ctxt.emit.emitIFFALSE(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitLABEL(labelCont) |
| (E) ctxt.emit.emitGOTO(labelStart) | |



31. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //5 trong đoạn mã trên

- (A) `ctxt.emit.emitGOTO(labelBreak)` (B) `ctxt.emit.emitIFFALSE(labelStart)`
(C) `ctxt.emit.emitLABEL(labelBreak)` (D) `ctxt.emit.emitLABEL(labelCont)`
(E) `ctxt.emit.emitGOTO(labelStart)`

32. Trình định thời (scheduled subprograms) thường được dùng trong

- (A) Lập trình song song (Parallel Programming)
(B) Lập trình hướng đối tượng (Object-Oriented Programming)
(C) Lập trình hàm (Functional Programming)
(D) Lập trình hướng sự kiện (Event-driven Programming)
(E) Lập trình thời gian thực (Real-time Programming)

33. Gọi `append` là hàm dùng để nối 2 danh sách thành 1 danh sách. Ví dụ `append([1,2,3],[4,5,6])` sẽ có kết quả là `[1,2,3,4,5,6]`. Hãy hiện thực hàm `append(a:List[Int],b:List[Int])` dùng hàm bậc cao (high-order function)?

- (A) `a.map(b)((x,y)=>x::y)` (B) `a.foldLeft(b)((x,y)=>x::y)`
(C) `a.foldRight(b)((x,y)=>x::y)` (D) `b.foldLeft(a)((x,y)=>x::y)`
(E) `b.foldRight(a)((x,y)=>x::y)`

Đoạn mã sau được dùng trong các câu 34– 35:

```
int *p = new int;  
void foo(int * r) {  
    delete r;  
}  
foo(p); //1  
*p = 2; //2
```

34. Hiện tượng gì xảy ra khi `p` được truyền cho `r` ở phát biểu //1 trong đoạn mã trên:

- (A) Con trỏ chưa khai báo (undeclared pointer)
(B) Bí danh (alias) (C) Khai báo trùng tên (redeclared)
(D) Tham chiếu treo (dangling reference) (E) Rác (garbage)

35. Hiện tượng gì xảy ra khi thực thi phát biểu //2 trong đoạn mã trên:

- (A) Tham chiếu treo (B) Bí danh (C) Đa hình (polymorphism)
(D) Rác (E) Con trỏ chưa khai báo

Đoạn mã sau được dùng cho các câu 36–41.

Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:

```
int A[5] = {1,3,5,7,9}; // index of A starts from 0  
int j = 0;  
int n = 5;  
int sumAndIncrease(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 = sumAndIncrease(A[j], j);
    printf("a = %i\n", s); //1
    printf(" %i %i %i %i %i\n", A[0], A[1], A[2], A[3], A[4]); //2
}
```

36. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- ☒ (A) 5 (B) 10 (C) 15 (D) 25 (E) Khác

37. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 2 4 6 8 10 ☒ (C) 6 3 5 7 9 (D) 5 3 5 7 9 (E) Khác

38. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 (B) 10 (C) 15 (D) 25 ☒ (E) Khác

39. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 ☒ (B) 2 4 6 8 10 (C) 6 3 5 7 9 (D) 5 3 5 7 9 (E) Khác

40. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 (B) 15 (C) 20 ☒ (D) 25 (E) Khác

41. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 ☒ (B) 2 4 6 8 10 (C) 6 3 5 7 9 (D) 5 3 5 7 9 (E) Khác

42. Một biến trên ngôn ngữ Javascript có thể nhận những giá trị thuộc các kiểu khác nhau. Ví dụ:

```
var x = "def";
```

```
x = 1;
```

Ràng buộc giữa một biến và kiểu của nó trên ngôn ngữ Javascript xảy ra trong thời gian

- (A) Hiện thực (implementation) ☒ (B) Chạy (running)
(C) Lập trình (programming) (D) Dịch (compiling)
(E) Khởi động (loading)

43. Cho $X = \{a, b\}$. Chọn biểu thức chính quy (regular expression) mô tả ngôn ngữ chứa bất kỳ chuỗi nào được tạo bởi các ký tự trên tập X nhưng KHÔNG chứa chuỗi có 2 ký tự a liên tiếp.

- (A) $a|(abb^*)^*$ (B) $(b^*ab^*)^*$ (C) $b^*(abb^*)^*$ ☒ (D) $b^*a?(bb^*a)^*b^*$
(E) $b^*ab^*ab^*$

44. Kiểu *union* của ngôn ngữ C++ cho phép ghi vào ở thành phần này nhưng đọc lên ở thành phần khác như trong ví dụ sau:

```
union {  
    int songuyen;  
    char kytu[2];  
} x;  
x.songuyen = 12;  
cout << x.kytu[0] << " " << x.kytu[1] << endl;
```

Đặc tính này của ngôn ngữ C++ gây ảnh hưởng XẤU NHẤT đối với

- (A) tính trực giao (orthogonality) (B) tính đơn giản (simplicity)
(C) tính dễ viết (writability) (D) tính tin cậy (reliability)
(E) chi phí (cost)

45. Cơ chế gọi chương trình con nào mà điều khiển có thể chuyển vào một vị trí ở giữa chương trình con được gọi thay vì chuyển vào đầu chương trình con được gọi?

- (A) Trình định thời (Scheduled Subprogram)
(B) Gọi trở về đơn giản (Simple Call - Return)
(C) Gọi đệ qui (Recursive call)
(D) Biến cố - Xử lý biến cố (Exception)
(E) Trình cộng hành (Coroutine)

46. Để thực hiện tìm kiếm một chuỗi (String) trên nhiều loại danh sách khác nhau, hãy chọn giải pháp thích hợp nhất để điền vào chỗ trống trong định nghĩa hàm sau:

```
def lookup[T](x:String, lst:List[T], f:T=>String):Option[T] =  
    lst match {  
        case List() => None  
        case h::t => _____  
    }
```

- (A) if (h == x) Some(h) else lookup(x,t,f)
(B) if (f(h) == x) Some(h) else lookup(x,t,f)
(C) if (h == f(x) Some(h) else lookup(x,t,f)
(D) if (f(h) == x) Some(f(h)) else lookup(x,t,f)
(E) if (h == f(x)) Some(f(h)) else lookup(x,t,f)

47. Khi cần khai thác đặc tính của kiểu union (các thành phần loại trừ lẫn nhau), làm thế nào để hiện thực trên những ngôn ngữ lập trình hướng đối tượng như Java (không có kiểu union)? Cụ thể làm thế nào để hiện thực biến `z` kiểu union sau trên những ngôn ngữ này?

```
union {  
    string loikhen;  
    string loiche;  
} z;
```

- (A) Định nghĩa lớp cha `A` và hai lớp con `B` (có thuộc tính `loikhen`) và `C` (có thuộc tính `loiche`), và `z` có kiểu `A`
- (B) Định nghĩa một lớp `A` có 2 thuộc tính `loikhen` và `loiche`; và `z` có kiểu `A`
- (C) Định nghĩa lớp `A` có thuộc tính `loikhen` và lớp `B` (con của `A`) có thuộc tính `loiche`, và `z` có kiểu `A`
- (D) Định nghĩa lớp `A` có thuộc tính `loikhen` và lớp `B` (con của `A`) có thuộc tính `loiche`, và `z` có kiểu `B`
- ☒ (E) Định nghĩa lớp trừu tượng (abstract class) `A` và 2 lớp con `B` (có thuộc tính `loikhen`) và `C` (có thuộc tính `loiche`), và `z` có kiểu `A`

II. Phần câu hỏi tự luận: (dành cho tất cả sinh viên) (2 điểm)

48. Trên những ngôn ngữ lập trình cho phép rút ngắn tính toán (short-circuit evaluation) biểu thức luận lý, biểu thức sau sẽ không gây ra lỗi khi thực thi.

$((a \neq 0) \ \&\& \ ((b / a) > 5))$

Hãy dùng **if then else** để mô phỏng quá trình tính toán của biểu thức luận lý trên?

Lời giải. if $((a \neq 0)$ then if $((b / a) > 8)$ then ...

Giải pháp đề xuất đúng: 2

49. Hãy nêu những giải pháp đã được sử dụng để tránh tham chiếu treo (dangling reference) trong trường hợp trả về hàm như sau:

```
void->int F() {  
    int x = 1;  
    int g() {  
        return x + 1;  
    }  
    return g ;  
}  
void->int gg = F();  
int z = gg();
```




50. 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

Lời giải. $(+ a (- (* b c d) e f))$

Nếu viết đúng biểu thức dạng tiền tố Cambridge Polish: 1

Nếu viết đúng và có số () ít nhất: 1

III. Phần bài tập lớn: (dành cho tất cả sinh viên)

51. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện kiểm tra kiểu. Hàm `visitBlock` sẽ thực hiện chức năng xây dựng bảng danh hiệu (symbol table) và truyền bảng này khi visit các phát biểu trong khối. Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm. Nhắc lại AST của Block được khai báo như sau:

```
case class Block(val decls:List[Decl],val stmts:List[Stmt]) extends Stmt
```

52. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện sinh mã cho một phát biểu If. Nhắc lại AST của If được khai báo như sau:

```
case class If(val expr:Expr, val thenStmt:Stmt, val elseStmt:Option[Stmt]) extends Stmt
```

Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm.



IV. Phần KSTN: (dành cho sinh viên KSTN)

53. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện kiểm tra lỗi `FunctionNotReTurn`? Yêu cầu giải thích ý tưởng thực hiện thể hiện qua mã của hàm?
54. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện sinh mã cho một khối (block)? Yêu cầu giải thích ý tưởng thực hiện sinh mã cho các khai báo array trong một khối?

Question Distribution

Content	Level 1	Level 2	Level 3	Level 4
Introduction		1		
Lexical	1	4	5	
Syntax	2	4	7	
AST		2		1
Total	3	11	12	1

HẾT

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



KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH
BỘ MÔN KHOA HỌC MÁY TÍNH

Họ và tên:.....
MSSV:.....

Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình

Thời gian: 120 phút

Ngày thi: 23-12-2015

☐ Sinh viên được phép sử dụng tài liệu

☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1152

- Sinh viên phải ghi tên và mã số sinh viên trên đề thi (**trang đầu và trang cuối**), giấy làm bài trắc nghiệm (giấy đỏ) và giấy làm bài. Sinh viên phải tô phần mã đề thi và mã số sinh viên trên giấy làm bài trắc nghiệm. Khi nộp bài, sinh viên phải nộp cả **đề thi, giấy làm bài trắc nghiệm và giấy làm bài**.
- Phần trắc nghiệm sẽ được chấm TỰ ĐỘNG trên giấy làm bài trắc nghiệm. Do đó, phần trắc nghiệm nếu làm trên đề thi sẽ KHÔNG được chấm.
- Đối với các câu hỏi phần trắc nghiệm, sinh viên chỉ chọn MỘT phương án đúng nhất.
- Đối với phần câu hỏi tự luận (phần II), sinh viên làm ngay trên đề thi, ở phần dành riêng ngay dưới mỗi câu hỏi.
- Đối với câu hỏi bài tập lớn (phần III và IV), sinh viên làm trên giấy làm bài.
- Sinh viên lớp đại trà làm 3 phần (I, II và III). Sinh viên lớp tài năng làm cả 4 phần (I, II, III và IV). Chỉ có phần I và II được dùng để tính điểm cuối kỳ, các phần III và IV để tính điểm bài tập lớn 2 và 3.

I. Phần câu hỏi trắc nghiệm:(8 điểm))

Phần trình bày sau dùng trong các câu hỏi 1– 2:

Cho mã giả của phát biểu *for var = expr1 to expr2 do body* như sau:

```
start:  var = expr1
        etemp = expr2
loop:   if (var >= etemp) goto out
        body
label1: var++
label2: goto loop
out:
```

- Nếu trong thân *body* có thực thi lệnh *continue* thì điều khiển sẽ chuyển đến thực thi phát biểu có nhãn là:

(A) out (B) label2 (C) start (D) loop (E) label1



2. Với hiện thực của phát biểu *for* như trên, phát biểu nào đặt trong thân phát biểu **for** dưới đây sẽ làm cho phát biểu này bị lặp mãi mãi?

```
n = 10; s = 0;  
for i = s to n do -----
```

- (A) $i = i - 1;$ (B) $s = s - 1;$ (C) $n = n + 1;$
(D) câu A và C đúng
(E) Không phát biểu nào trong các câu A, B và C làm phát biểu **for** lặp mãi mãi

3. Cơ chế gọi chương trình nào là cơ chế gọi chương trình con cơ bản của Lập trình hướng sự kiện (Event-driven Programming)

- (A) Biến cố - Xử lý biến cố (Exception)
(B) Trình định thời (Scheduled Subprogram)
(C) Gọi trở về đơn giản (Simple Call - Return)
(D) Gọi đệ qui (Recursive call)
(E) Trình cộng hành (Coroutine)

4. Cho biết kích thước của đối tượng dữ liệu **x** được khai báo như sau:
x: set of 1..16

- (A) 4 bytes (B) 16 bytes (C) 2 bytes (D) 4 bits (E) 1 byte

5. Cho khai báo sau được viết trên ngôn ngữ lập trình Ada:

```
type Shape is ( Circle , Triangle , Rectangle );  
type Colors is ( Red , Green , Blue );  
type Figure (Form: Shape) is record  
    Filled: Boolean;  
    Color: Colors;  
    case Form is  
        when Circle => Diameter: Float;  
        when Triangle =>  
            Leftside , Rightside: Integer;  
            Angle: Float;  
        when Rectangle => Side1 , Side2: Integer;  
    end case;  
end record;
```

Cho kích thước của các kiểu *Integer*, *Float*, *Boolean* và *Enumeration* lần lượt là 2, 4, 1, và 2, và kích thước cho phần mô tả kiểu (type description) là 0. Cho biết kích thước của một đối tượng kiểu **Figure**?

- (A) 15 (B) 21 (C) 13 (D) 23 (E) Khác

6. Để thực hiện tìm kiếm một chuỗi (String) trên nhiều loại danh sách khác nhau, hãy chọn giải pháp thích hợp nhất để điền vào chỗ trống trong định nghĩa hàm sau:

```
def lookup[T](x:String, lst:List[T], f:T=>String):Option[T] =
  lst match {
    case List() => None
    case h::t => _____
  }
```

- (A) if (h == x) Some(h) else lookup(x,t,f)
 (B) if (h == f(x)) Some(f(h)) else lookup(x,t,f)
 (C) if (f(h) == x) Some(h) else lookup(x,t,f)
 (D) if (h == f(x)) Some(h) else lookup(x,t,f)
 (E) if (f(h) == x) Some(f(h)) else lookup(x,t,f)

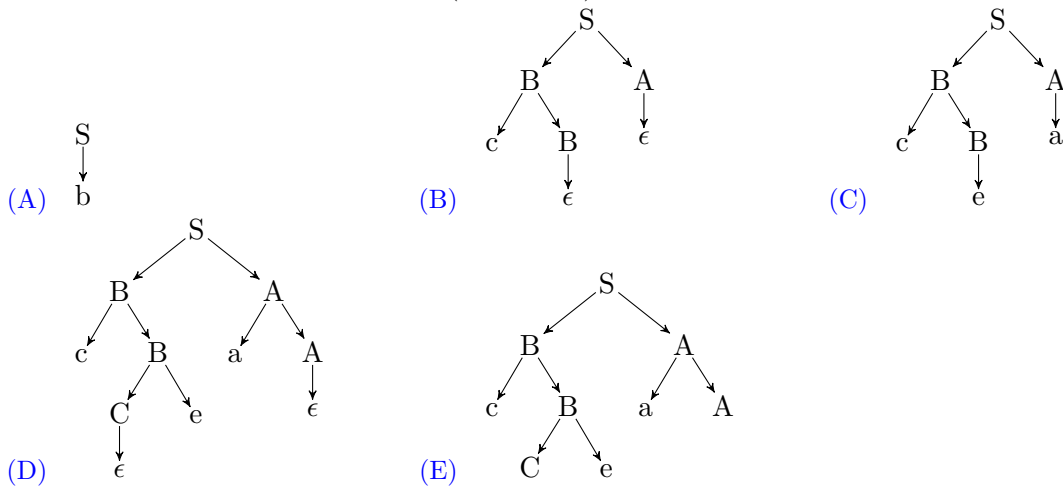
Cho văn phạm sau dùng cho các câu 7–8:

S → b | B A
 A → a A | ε
 B → C e | c B
 C → d C | ε

7. Chuỗi nào KHÔNG thuộc ngôn ngữ được mô tả bởi văn phạm trên?

- (A) e (B) cccaaa (C) b (D) eaaa (E) cccddddeaa

8. Chọn cây phân tích cú pháp (parse tree) cho chuỗi nhập: **cea**



Đoạn mã sau được dùng cho các câu 9–14.

Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:

```
int A[5] = {1,3,5,7,9}; // index of A starts from 0
int j = 0;
int n = 5;
int sumAndIncrease(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 = sumAndIncrease(A[j], j);
    printf("a = %i\n", s); //1
    printf(" %i %i %i %i %i\n", A[0], A[1], A[2], A[3], A[4]); //2
}
```

9. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?
(A) 25 (B) 5 (C) 10 (D) 15 (E) Khác
10. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?
(A) 5 3 5 7 9 (B) 1 3 5 7 9 (C) 2 4 6 8 10 (D) 6 3 5 7 9 (E) Khác
11. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?
(A) 25 (B) 5 (C) 10 (D) 15 (E) Khác
12. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?
(A) 5 3 5 7 9 (B) 1 3 5 7 9 (C) 2 4 6 8 10 (D) 6 3 5 7 9 (E) Khác
13. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?
(A) 25 (B) 5 (C) 15 (D) 20 (E) Khác
14. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?
(A) 5 3 5 7 9 (B) 1 3 5 7 9 (C) 2 4 6 8 10 (D) 6 3 5 7 9 (E) Khác
15. Kiểu *union* của ngôn ngữ C++ cho phép ghi vào ở thành phần này nhưng đọc lên ở thành phần khác như trong ví dụ sau:

```
union {
    int songuyen;
    char kytu[2];
} x;
x.songuyen = 12;
cout << x.kytu[0] << " " << x.kytu[1] << endl;
```

Đặc tính này của ngôn ngữ C++ gây ảnh hưởng XẤU NHẤT đối với

- (A) tính trực giao (orthogonality) (B) chi phí (cost)
(C) tính đơn giản (simplicity) (D) tính dễ viết (writability)
(E) tính tin cậy (reliability)

16. Cho biết mã Jasmin của biểu thức viết bằng ngôn ngữ C như sau:

$a * b - 2$

với a và b là 2 biến nguyên có chỉ số lần lượt là 0 và 1.

	iload_0	imul	iload_0	iload_0	iload_0
	iload_1	load_0	iload_1	imul	iload_1
(A)	imul	(B) iload_1	(C) iconst_2	(D) iload_1	(E) iconst_2
	iconst_2	isub	imul	isub	isub
	isub	iconst_2	isub	iconst_2	imul

17. Viết lại biểu thức trung tố (infix) sau sang dạng hậu tố (postfix) Polish? Giả sử là tất cả các phép toán đều có 2 toán hạng và có độ ưu tiên và tính kết hợp như trong ngôn ngữ lập trình C.

$a + b * c - d * e + f$

- (A) $a + b * c * d * e * - f +$ (B) $a * b * c * + d * e * - f +$ (C) $a * b + c * d - e * f +$
 (D) $a * b * c * + d - e * f * +$ (E) $a * b * c * + d * e * + f -$

Phần hướng dẫn này áp dụng cho các câu 18–22

Hãy viết thêm các đoạn mã cần thiết vào phương thức dưới đây để thực hiện sinh mã cho phát biểu *repeat*. Nhắc lại phát biểu *repeat* kết thúc quá trình lặp khi điều kiện của phát biểu trở thành **true**. AST của phát biểu **repeat** được định nghĩa như sau:

```
case class Repeat(val stmts: List[Stmt], val exp: Expr) extends Stmt
```

```
override def visitRepeat(ast: Repeat, o: Context) = {
  val ctxt = o.asInstanceOf[SubBody]
  ctxt.frame.enterLoop();
  val labelStart = ctxt.frame.getNewLabel()
  val labelBreak = ctxt.frame.getBreakLabel()
  val labelCont = ctxt.frame.getContinueLabel()
  val str1 = //1
  // sinh mã cho từng phát biểu trong thân của repeat
  val str2 = //2
  val str3 = //3
  // sinh mã cho biểu thức điều kiện
  val str4 = visit(ast.exp, ...)
  val str5 = //4
  val str6 = //5
  frame().exitLoop();
  str1 + str2 + str3 + str4 + str5 + str6
}
```

18. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //1 trong đoạn mã trên

- (A) "" (B) `ctxt.emit.emitGOTO(labelStart)`
 (C) `ctxt.emit.emitLABEL(labelStart)` (D) `ctxt.emit.emitLABEL(labelBreak)`
 (E) `ctxt.emit.emitLABEL(labelCont)`



19. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //2 trong đoạn mã trên

- (A) `visit(ast.stmts,o)`
- (B) `ast.stmts.foldLeft(")((x,y)=>y + visit(x,o))`
- (C) `ast.stmts.map(x=>visit(x,o))`
- (D) `ast.stmts.filter(x=>visit(x,o))`
- (E) `ast.stmts.foldLeft(")((x,y)=>x + visit(y,o))`

20. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //3 trong đoạn mã trên

- (A) `"`
- (B) `ctxt.emit.emitGOTO(labelStart)`
- (C) `ctxt.emit.emitIFFALSE(labelStart)`
- (D) `ctxt.emit.emitLABEL(labelBreak)`
- (E) `ctxt.emit.emitLABEL(labelCont)`

21. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //4 trong đoạn mã trên

- (A) `ctxt.emit.emitGOTO(labelBreak)`
- (B) `ctxt.emit.emitGOTO(labelStart)`
- (C) `ctxt.emit.emitIFFALSE(labelStart)`
- (D) `ctxt.emit.emitLABEL(labelBreak)`
- (E) `ctxt.emit.emitLABEL(labelCont)`

22. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //5 trong đoạn mã trên

- (A) `ctxt.emit.emitGOTO(labelBreak))`
- (B) `ctxt.emit.emitGOTO(labelStart))`
- (C) `ctxt.emit.emitIFFALSE(labelStart))`
- (D) `ctxt.emit.emitLABEL(labelBreak))`
- (E) `ctxt.emit.emitLABEL(labelCont))`

23. Cho lớp(class) A là lớp cha (superclass) của lớp B và B là lớp cha của hai lớp C và D. Trong lớp A, có khai báo phương thức thực thể (instance method) **foo** và phương thức này bị ghi đè(override) trên tất cả các lớp B, C và D. Cho biến x được khai báo như sau:
B x;

Qui ước viết $func_T$ là phương thức func được khai báo trong lớp T, cho biết tập của phương thức đích của cuộc gọi sau: **x.foo();**

- (A) $\{foo_C, foo_D\}$
- (B) $\{foo_B, foo_C, foo_D\}$
- (C) $\{foo_A, foo_B, foo_C, foo_D\}$
- (D) $\{foo_B\}$
- (E) $\{foo_A, foo_B\}$

24. Liệt kê theo thứ tự chiều dài tăng dần ít nhất 5 chuỗi ngắn nhất (nếu có nhiều hơn 5 chuỗi) của ngôn ngữ được mô tả bởi biểu thức chính quy (regular expression) sau:
 $ab^*(a|b)?a$

- (A) aa, aba, aaa, abaa, aaba
- (B) aa, aba, aaa, abaa, aaaa
- (C) ϵ , a, aa, aba, abaa
- (D) aa, aba, aaa, abaa, abba
- (E) a, aa, aba, aaa, aaba

Cho văn phạm của một biểu thức sau dùng cho các câu 25–27:

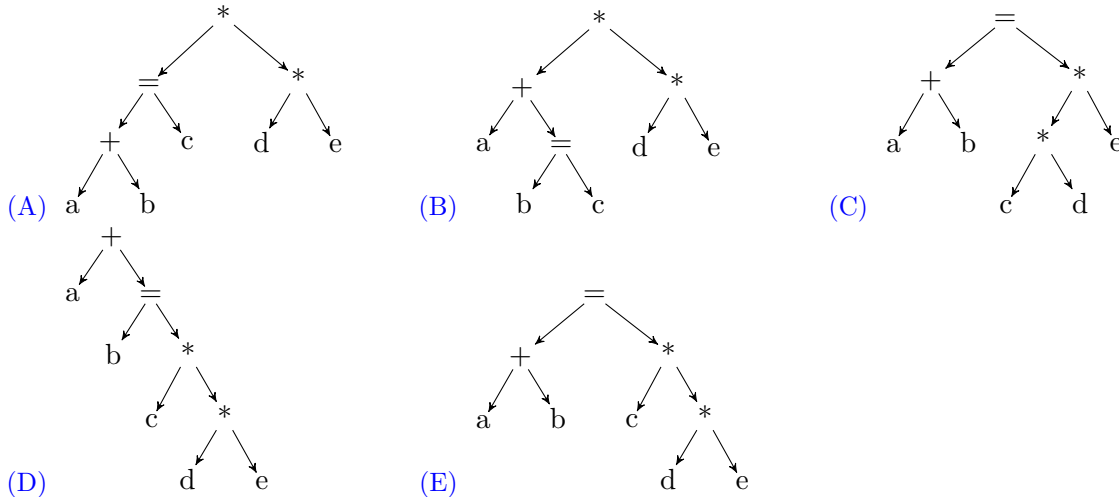
$exp \rightarrow term '=' exp \mid term$
 $term \rightarrow term '+' fact \mid term '>' fact \mid fact$
 $fact \rightarrow ope '*' ope \mid ope$
 $ope \rightarrow '(' exp ')' \mid ID$

với ID là một danh hiệu.

25. Chọn biểu thức được viết đúng văn phạm, có số lượng dấu '(' và ')' ít nhất mà tương đương với biểu thức sau: $(a + (b = ((c * d) * e))) > f$

- (A) $(a + (b = (c * d) * e)) > f$ (B) $a + (b = c * d * e) > f$
 (C) $(a + (b = ((c * d) * e))) > f$ (D) $a + b = c * d * e > f$
 (E) $a + (b = (c * d) * e) > f$

26. Vẽ AST cho biểu thức sau được viết dựa vào văn phạm trên: $a + b = c * (d * e)$



27. Cho cấu trúc AST được khai báo trên Scala như sau:

```
trait Exp
case class Bin(op:String,e1:Exp,e2:Exp) extends Exp
case class Id(i:String) extends Exp
```

Chọn AST cho biểu thức sau: $a + b > c = d * e = f$

- (A) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Bin("=",Id("e"),Id("f"))))`
 (B) `Bin("=",Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("*",Id("d"),Id("e"))),Id("f"))`
 (C) `Bin("=",Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Id("e"))),Id("f"))`
 (D) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("=",Bin("*",Id("d"),Id("e")),Id("f"))`
 (E) `Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("=",Bin("*",Id("d"),Id("e")),Id("f"))`

28. Cơ chế gọi chương trình con nào mà điều khiển có thể chuyển vào một vị trí ở giữa chương trình con được gọi thay vì chuyển vào đầu chương trình con được gọi?

- (A) Trình định thời (Scheduled Subprogram)
 (B) Trình cộng hành (Coroutine)
 (C) Gọi trở về đơn giản (Simple Call - Return)
 (D) Gọi đệ qui (Recursive call)
 (E) Biến cố - Xử lý biến cố (Exception)

29. Giả sử chương trình có một lỗi văn phạm nằm sau lệnh **print**. Khi em nhấn nút Run (để dịch và chạy chương trình) trên trình soạn thảo, kết quả được in ra bởi lệnh **print** trước khi lỗi văn phạm được báo. Hỏi chế độ dịch của trình soạn thảo là gì?

- (A) trình liên kết (link editor) (C) trình biên dịch (compiler)
 (B) trình biên dịch động(just-in-time compiler) (D) trình thông dịch(interpreter)
 (E) trình hợp ngữ (assembler)



30. Cho giá trị ban đầu của biến **c** là 3, cho biết những giá trị **có thể có** của biến **a** sau khi thực hiện phép gán sau trên ngôn ngữ lập trình C?

$a = c * (c = 5);$

- (A) 15, 25, 9 (B) 15, 25 (C) 15 (D) 25 (E) 9

31. Đoạn code viết bằng Scala nào dưới đây có thể dùng như một ví dụ minh họa cho khái niệm Hàm Currying

- (A) `def add1(n:Int) = n + 1`
(B) `List(1,2,3).filter(_ > 1)`
(C) `List(1,2,3).foldLeft(0)(_ + _)`
(D) `def add(n:Int) = (x:Int) => n + x; val add2 = add(2)`
(E) `def add(n:Int)(x:Int) = n + x; val add2 = add (2) _`

Đoạn mã sau được dùng trong các câu 32– 33:

```
int *p = new int;  
void foo(int * r) {  
    delete r;  
}  
foo(p); //1  
*p = 2; //2
```

32. Hiện tượng gì xảy ra khi **p** được truyền cho **r** ở phát biểu //1 trong đoạn mã trên:

- (A) Con trỏ chưa khai báo (undeclared pointer) (B) Rác (garbage)
(C) Bí danh (alias) (D) Khai báo trùng tên (redeclared)
(E) Tham chiếu treo (dangling reference)

33. Hiện tượng gì xảy ra khi thực thi phát biểu //2 trong đoạn mã trên:

- (A) Tham chiếu treo (B) Con trỏ chưa khai báo (C) Bí danh
(D) Đa hình (polymorphism) (E) Rác

34. Khi cần khai thác đặc tính của kiểu union (các thành phần loại trừ lẫn nhau), làm thế nào để hiện thực trên những ngôn ngữ lập trình hướng đối tượng như Java (không có kiểu union)? Cụ thể làm thế nào để hiện thực biến **z** kiểu union sau trên những ngôn ngữ này?

```
union {  
    string loikhen;  
    string loiche;  
} z;
```

- (A) Định nghĩa lớp cha A và hai lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A
(B) Định nghĩa lớp trừu tượng (abstract class) A và 2 lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A
(C) Định nghĩa một lớp A có 2 thuộc tính loikhen và loiche; và z có kiểu A
(D) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu A
(E) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu B

35. Trình định thời (scheduled subprograms) thường được dùng trong

- (A) Lập trình song song (Parallel Programming)
- (B) Lập trình thời gian thực (Real-time Programming)
- (C) Lập trình hướng đối tượng (Object-Oriented Programming)
- (D) Lập trình hàm (Functional Programming)
- (E) Lập trình hướng sự kiện (Event-driven Programming)

36. Cho biết mã Jasmin của phát biểu gán sau:

$$a[i] = a[i] + 2$$

với i kiểu nguyên có chỉ số là 3 và a là kiểu dãy nguyên có chỉ số là 1.

- | | | | | | | | |
|-----|----------|-----|----------|-----|----------|-----|----------|
| | iastore | | aload_1 | | aload_1 | | aload_1 |
| | iadd | | iload_3 | | aload_1 | | dup |
| | iaload | | iastore | | iload_3 | | iload_3 |
| (A) | aload_1 | (B) | aload_1 | (C) | iconst_2 | (D) | iaload |
| | iload_3 | | iload_3 | | iadd | | iconst_2 |
| | dup | | iconst_2 | | iastore | | iadd |
| | iconst_2 | | iadd | | ... | | iastore |
| | ... | | ... | | ... | | ... |

37. Chọn phát biểu ĐÚNG trong các phát biểu về dãy (array) sau:

- (A) Trên C, kích thước của dãy có thể tăng thêm sau khi dãy đã được tạo ra
- (B) Truy xuất một phần tử bên ngoài một dãy Java không được phép, sẽ lập tức gây ra lỗi
- (C) Truy xuất một phần tử bên ngoài một dãy C không được phép, sẽ lập tức gây ra lỗi
- (D) Trên C và C++, kích thước của một dãy được lưu trữ để kiểm tra động
- (E) Trên Java, kích thước của một dãy KHÔNG được lưu trữ để kiểm tra động

Đoạn code sau dùng cho các câu 38–40

```
int p;
int* foo(int x) {
    static int q;
    int *s = new int;
    switch (x) {
        case 1: return &p;
        case 2: return &q;
        case 3: return &x;
        case 4: return s;
        default: return foo(x-1);
    }
}
```

38. Phát biểu nào sẽ gây ra lỗi tham chiếu treo (dangling reference) khi thực thi

- (A) return s
- (B) return &p
- (C) return &q
- (D) return &x
- (E) Không phát biểu nào gây ra lỗi trên khi thực thi

39. Trong đoạn mã trên, đối tượng dữ liệu nào có thể trở thành rác (garbage)

- (A) trỏ đến bởi s
- (B) p
- (C) q
- (D) x
- (E) Không có đối tượng nào có thể trở thành rác



40. Khi hàm **foo** được gọi đệ quy (recursive) thì các bản hoạt động của **foo** dùng chung những đối tượng dữ liệu nào?

- (A) q và s (B) x, q và s (C) p, q, x và s (D) p (E) p và q

41. Gọi **append** là hàm dùng để nối 2 danh sách thành 1 danh sách. Ví dụ **append([1,2,3],[4,5,6])** sẽ có kết quả là **[1,2,3,4,5,6]**. Hãy hiện thực hàm **append(a:List[Int],b:List[Int])** dùng hàm bậc cao (high-order function)?

- (A) **a.map(b)((x,y)=>x::y)** (B) **b.foldRight(a)((x,y)=>x::y)**
(C) **a.foldLeft(b)((x,y)=>x::y)** (D) **a.foldRight(b)((x,y)=>x::y)**
(E) **b.foldLeft(a)((x,y)=>x::y)**

42. Một biến trên ngôn ngữ Javascript có thể nhận những giá trị thuộc các kiểu khác nhau. Ví dụ:

`var x = "def";`

`x = 1;`

Ràng buộc giữa một biến và kiểu của nó trên ngôn ngữ Javascript xảy ra trong thời gian

- (A) Hiện thực (implementation) (B) Khởi động (loading)
(C) Chạy (running) (D) Lập trình (programming)
(E) Dịch (compiling)

Đoạn code sau, được viết trên ngôn ngữ dùng **qui tắc tầm vực tĩnh (static-scope rule)**, áp dụng cho các câu 43–46

```
procedure main(){
  var a, b, c:integer; //1
  procedure sub1(a: integer) { //2
    procedure sub3();
    procedure sub2() {
      var a, c : real;//3
      sub3();
    }
    procedure sub3() {
      a // use a
    }
    sub2();
  }
  sub1(3);
}
```

43. Môi trường tham khảo tĩnh (static referencing environment) của thủ tục **sub3** KHÔNG chứa

- (A) sub2 (B) a ở //1 (C) a ở //2 (D) sub1 (E) sub3

44. Giả sử chuỗi gọi là **main → sub1 → sub2 → sub3**, tham khảo đến **a** trong **sub3** ứng với khai báo

- (A) Báo lỗi a chưa khai báo (B) a trong main (//1) (C) a trong sub1 (//2)
(D) a trong sub2 (//3) (E) a trong sub3

45. Giả sử chuỗi gọi là **main → sub1 → sub2 → sub3**, tham khảo đến **a** trong sub3 **khi thực thi** ứng với kết hợp của **a** trong bản hoạt động của

- (A) main (B) Báo lỗi không tìm thấy a (C) sub1 (D) sub2
(E) sub3



50. 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

III. Phần bài tập lớn: (dành cho tất cả sinh viên)

51. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện kiểm tra kiểu. Hàm `visitBlock` sẽ thực hiện chức năng xây dựng bảng danh hiệu (symbol table) và truyền bảng này khi visit các phát biểu trong khối. Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm. Nhắc lại AST của Block được khai báo như sau:

```
case class Block(val decl:List[Decl],val stmts:List[Stmt]) extends Stmt
```

52. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện sinh mã cho một phát biểu If. Nhắc lại AST của If được khai báo như sau:

```
case class If(val expr:Expr, val thenStmt:Stmt, val elseStmt:Option[Stmt]) extends Stmt
```

Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm.

IV. Phần KSTN: (dành cho sinh viên KSTN)

53. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện kiểm tra lỗi FunctionNotReTurn? Yêu cầu giải thích ý tưởng thực hiện thể hiện qua mã của hàm?

54. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện sinh mã cho một khối (block)? Yêu cầu giải thích ý tưởng thực hiện sinh mã cho các khai báo array trong một khối?

HẾT

Họ và tên:.....

Điểm:

MSSV:.....



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



KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH
BỘ MÔN KHOA HỌC MÁY TÍNH

ĐÁP ÁN cho Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình

Thời gian: 120 phút

☐ Sinh viên được phép sử dụng tài liệu

☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1152

I. Phần câu hỏi trắc nghiệm:(8 điểm))

Phần trình bày sau dùng trong các câu hỏi 1– 2:

Cho mã giả của phát biểu *for* $var = expr1$ to $expr2$ do *body* như sau:

```
start:  var = expr1
        etemp = expr2
loop:   if (var >= etemp) goto out
        body
label1: var++
label2: goto loop
out:
```

1. Nếu trong thân *body* có thực thi lệnh *continue* thì điều khiển sẽ chuyển đến thực thi phát biểu có nhãn là:

(A) out (B) label2 (C) start (D) loop E label1

2. Với hiện thực của phát biểu *for* như trên, phát biểu nào đặt trong thân phát biểu **for** dưới đây sẽ làm cho phát biểu này bị lặp mãi mãi?

```
n = 10; s = 0;
for i = s to n do _____
```

A i = i - 1; (B) s = s - 1; (C) n = n + 1;
(D) câu A và C đúng
(E) Không phát biểu nào trong các câu A, B và C làm phát biểu **for** lặp mãi mãi

3. Cơ chế gọi chương trình nào là cơ chế gọi chương trình con cơ bản của Lập trình hướng sự kiện (Event-driven Programming)

A Biến cố - Xử lý biến cố (Exception)
(B) Trình định thời (Scheduled Subprogram)
(C) Gọi trở về đơn giản (Simple Call - Return)
(D) Gọi đệ qui (Recursive call)
(E) Trình cộng hành (Coroutine)

4. Cho biết kích thước của đối tượng dữ liệu **x** được khai báo như sau:
x: set of 1..16

(A) 4 bytes (B) 16 bytes C 2 bytes (D) 4 bits (E) 1 byte



5. Cho khai báo sau được viết trên ngôn ngữ lập trình Ada:

```

type Shape is ( Circle , Triangle , Rectangle );
type Colors is ( Red, Green, Blue );
type Figure (Form: Shape) is record
    Filled: Boolean;
    Color: Colors;
    case Form is
        when Circle => Diameter: Float;
        when Triangle =>
            Leftside, Rightside: Integer;
            Angle: Float;
        when Rectangle => Side1, Side2: Integer;
    end case;
end record;

```

Cho kích thước của các kiểu *Integer*, *Float*, *Boolean* và *Enumeration* lần lượt là 2, 4, 1, và 2, và kích thước cho phần mô tả kiểu (type description) là 0. Cho biết kích thước của một đối tượng kiểu **Figure**?

- (A) 15 (B) 21 C 13 (D) 23 (E) Khác

6. Để thực hiện tìm kiếm một chuỗi (String) trên nhiều loại danh sách khác nhau, hãy chọn giải pháp thích hợp nhất để điền vào chỗ trống trong định nghĩa hàm sau:

```

def lookup[T](x:String, lst:List[T], f:T=>String):Option[T] =
  lst match {
    case List() => None
    case h::t => _____
  }

```

- (A) if (h == x) Some(h) else lookup(x,t,f)
 (B) if (h == f(x)) Some(f(h)) else lookup(x,t,f)
C if (f(h) == x) Some(h) else lookup(x,t,f)
 (D) if (h == f(x) Some(h) else lookup(x,t,f)
 (E) if (f(h) == x) Some(f(h)) else lookup(x,t,f)

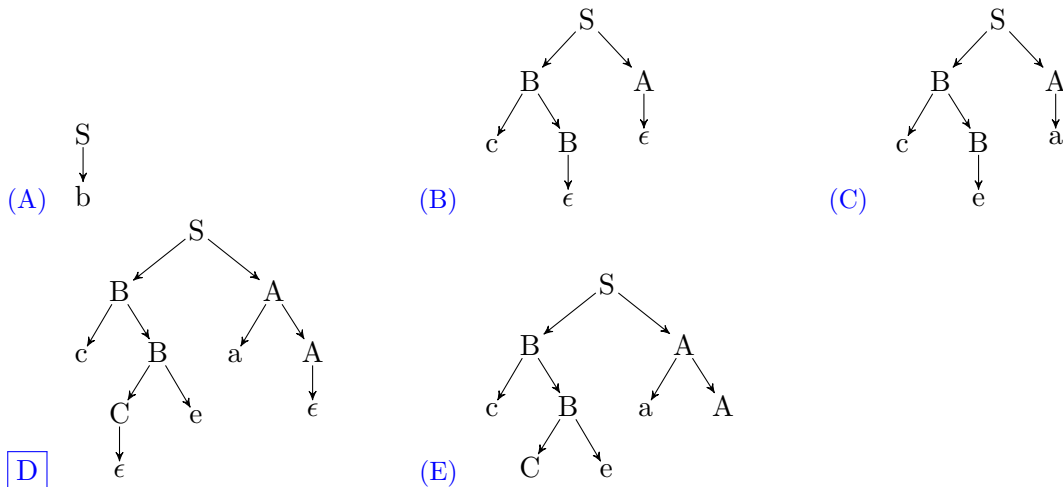
Cho văn phạm sau dùng cho các câu 7–8:

S	→	b		B A
A	→	a A		ε
B	→	C e		c B
C	→	d C		ε

7. Chuỗi nào KHÔNG thuộc ngôn ngữ được mô tả bởi văn phạm trên?

- (A) e B cccaaa (C) b (D) eaaa (E) cccddddeaa

8. Chọn cây phân tích cú pháp (parse tree) cho chuỗi nhập: **cea**



Đoạn mã sau được dùng cho các câu 9–14.

Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:

```
int A[5] = {1,3,5,7,9}; // index of A starts from 0
int j = 0;
int n = 5;
int sumAndIncrease(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 = sumAndIncrease(A[j], j);
    printf("a = %i\n", s); //1
    printf(" %i %i %i %i %i\n", A[0], A[1], A[2], A[3], A[4]); //2
}
```

9. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 25 ☒ (B) 5 (C) 10 (D) 15 (E) Khác

10. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 5 3 5 7 9 (B) 1 3 5 7 9 (C) 2 4 6 8 10 ☒ (D) 6 3 5 7 9 (E) Khác

11. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 25 (B) 5 (C) 10 (D) 15 ☒ (E) Khác



12. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 5 3 5 7 9 (B) 1 3 5 7 9 ☒ (C) 2 4 6 8 10 (D) 6 3 5 7 9 (E) Khác

13. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- ☒ (A) 25 (B) 5 (C) 15 (D) 20 (E) Khác

14. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 5 3 5 7 9 (B) 1 3 5 7 9 ☒ (C) 2 4 6 8 10 (D) 6 3 5 7 9 (E) Khác

15. Kiểu *union* của ngôn ngữ C++ cho phép ghi vào ở thành phần này nhưng đọc lên ở thành phần khác như trong ví dụ sau:

```
union {
    int songuyen;
    char kytu[2];
} x;
x.songuyen = 12;
cout << x.kytu[0] << " " << x.kytu[1] << endl;
```

Đặc tính này của ngôn ngữ C++ gây ảnh hưởng XẤU NHẤT đối với

- (A) tính trực giao (orthogonality) (B) chi phí (cost)
 (C) tính đơn giản (simplicity) (D) tính dễ viết (writability)
☒ (E) tính tin cậy (reliability)

16. Cho biết mã Jasmin của biểu thức viết bằng ngôn ngữ C như sau:

$a * b - 2$

với a và b là 2 biến nguyên có chỉ số lần lượt là 0 và 1.

- | | | | | | | | | | |
|---|---|-----|--|-----|---|-----|---|-----|---|
| <input checked="" type="checkbox"/> (A) | iload_0
iload_1
imul
iconst_2
isub
... | (B) | imul
load_0
iload_1
isub
iconst_2
... | (C) | iload_0
iload_1
iconst_2
imul
isub
... | (D) | iload_0
imul
iload_1
isub
iconst_2
... | (E) | iload_0
iload_1
iconst_2
isub
imul
... |
|---|---|-----|--|-----|---|-----|---|-----|---|

17. Viết lại biểu thức trung tố (infix) sau sang dạng hậu tố (postfix) Polish? Giả sử là tất cả các phép toán đều có 2 toán hạng và có độ ưu tiên và tính kết hợp như trong ngôn ngữ lập trình C.

$a + b * c - d * e + f$

- (A) $a + b c * d e * - f +$ ☒ (B) $a b c * + d e * - f +$ (C) $a b + c * d - e * f +$
 (D) $a b c * + d - e f * +$ (E) $a b c * + d e * + f -$

Phần hướng dẫn này áp dụng cho các câu 18–22

Hãy viết thêm các đoạn mã cần thiết vào phương thức dưới đây để thực hiện sinh mã cho phát biểu *repeat*. Nhắc lại phát biểu *repeat* kết thúc quá trình lặp khi điều kiện của phát biểu trở thành **true**. AST của phát biểu **repeat** được định nghĩa như sau:

```
case class Repeat(val stmts:List[Stmt],val exp:Expr) extends Stmt
```



```
override def visitRepeat(ast: Repeat, o: Context) = {  
  val ctxt = o.asInstanceOf[SubBody]  
  ctxt.frame.enterLoop();  
  val labelStart = ctxt.frame.getNewLabel()  
  val labelBreak = ctxt.frame.getBreakLabel()  
  val labelCont = ctxt.frame.getContinueLabel()  
  val str1 = //1  
  // sinh mã cho từng phát biểu trong thân của repeat  
  val str2 = //2  
  val str3 = //3  
  // sinh mã cho biểu thức điều kiện  
  val str4 = visit(ast.exp,...)  
  val str5 = //4  
  val str6 = //5  
  frame().exitLoop();  
  str1 + str2 + str3 + str4 + str5 + str6  
}
```

18. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //1 trong đoạn mã trên

- | | |
|-------------------------------------|-------------------------------------|
| (A) "" | (B) ctxt.emit.emitGOTO(labelStart) |
| (C) ctxt.emit.emitLABEL(labelStart) | (D) ctxt.emit.emitLABEL(labelBreak) |
| (E) ctxt.emit.emitLABEL(labelCont) | |

19. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //2 trong đoạn mã trên

- (A) visit(ast.stmts,o)
(B) ast.stmts.foldLeft(")((x,y)=>y + visit(x,o))
(C) ast.stmts.map(x=>visit(x,o))
(D) ast.stmts.filter(x=>visit(x,o))
(E) ast.stmts.foldLeft(")((x,y)=>x + visit(y,o))

20. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //3 trong đoạn mã trên

- | | |
|---------------------------------------|-------------------------------------|
| (A) "" | (B) ctxt.emit.emitGOTO(labelStart) |
| (C) ctxt.emit.emitIFFALSE(labelStart) | (D) ctxt.emit.emitLABEL(labelBreak) |
| (E) ctxt.emit.emitLABEL(labelCont) | |

21. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //4 trong đoạn mã trên

- | | |
|---------------------------------------|-------------------------------------|
| (A) ctxt.emit.emitGOTO(labelBreak) | (B) ctxt.emit.emitGOTO(labelStart) |
| (C) ctxt.emit.emitIFFALSE(labelStart) | (D) ctxt.emit.emitLABEL(labelBreak) |
| (E) ctxt.emit.emitLABEL(labelCont) | |

22. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //5 trong đoạn mã trên

- | | |
|--|--------------------------------------|
| (A) ctxt.emit.emitGOTO(labelBreak)) | (B) ctxt.emit.emitGOTO(labelStart)) |
| (C) ctxt.emit.emitIFFALSE(labelStart)) | (D) ctxt.emit.emitLABEL(labelBreak)) |
| (E) ctxt.emit.emitLABEL(labelCont)) | |

23. Cho lớp(class) A là lớp cha (superclass) của lớp B và B là lớp cha của hai lớp C và D. Trong lớp A, có khai báo phương thức thực thể (instance method) **foo** và phương thức này bị ghi đè(override) trên tất cả các lớp B, C và D. Cho biến x được khai báo như sau:
B x;

Qui ước viết $func_T$ là phương thức func được khai báo trong lớp T, cho biết tập của phương thức đích của cuộc gọi sau: **x.foo();**

- (A) $\{foo_C, foo_D\}$ (B) $\{foo_B, foo_C, foo_D\}$ (C) $\{foo_A, foo_B, foo_C, foo_D\}$
(D) $\{foo_B\}$ (E) $\{foo_A, foo_B\}$

24. Liệt kê theo thứ tự chiều dài tăng dần ít nhất 5 chuỗi ngắn nhất (nếu có nhiều hơn 5 chuỗi) của ngôn ngữ được mô tả bởi biểu thức chính quy (regular expression) sau:
 $ab^*(a|b)?a$

- (A) aa, aba, aaa, abaa, aaba (B) aa, aba, aaa, abaa, aaaa (C) ϵ , a, aa, aba, abaa
(D) aa, aba, aaa, abaa, abba (E) a, aa, aba, aaa, aaba

Cho văn phạm của một biểu thức sau dùng cho các câu 25–27:

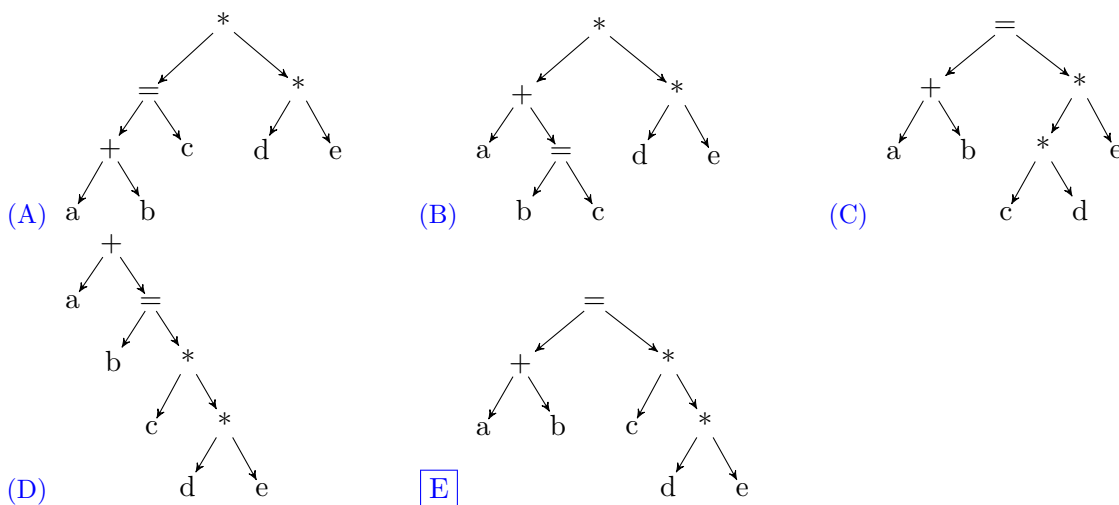
exp \rightarrow term '=' exp | term
term \rightarrow term '+' fact | term '>' fact | fact
fact \rightarrow ope '**' ope | ope
ope \rightarrow '(' exp ')' | ID

với ID là một danh hiệu.

25. Chọn biểu thức được viết đúng văn phạm, có số lượng dấu '(' và ')' ít nhất mà tương đương với biểu thức sau: **$(a + (b = ((c * d) * e))) > f$**

- (A) $(a + (b = (c * d) * e)) > f$ (B) $a + (b = c * d * e) > f$
(C) $(a + (b = ((c * d) * e))) > f$ (D) $a + b = c * d * e > f$
(E) $a + (b = (c * d) * e) > f$

26. Vẽ AST cho biểu thức sau được viết dựa vào văn phạm trên: **$a + b = c * (d * e)$**





27. Cho cấu trúc AST được khai báo trên Scala như sau:

```
trait Exp
case class Bin(op:String,e1:Exp,e2:Exp) extends Exp
case class Id(i:String) extends Exp
```

Chọn AST cho biểu thức sau: $a + b > c = d * e = f$

- (A) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Bin("=",Id("e"),Id("f"))))`
- (B) `Bin("=",Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("*",Id("d"),Id("e"))),Id("f"))`
- (C) `Bin("=",Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Id("e"))),Id("f"))`
- ☒ (D) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("=",Bin("*",Id("d"),Id("e"))),Id("f"))`
- (E) `Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("=",Bin("*",Id("d"),Id("e"))),Id("f"))`

28. Cơ chế gọi chương trình con nào mà điều khiển có thể chuyển vào một vị trí ở giữa chương trình con được gọi thay vì chuyển vào đầu chương trình con được gọi?

- (A) Trình định thời (Scheduled Subprogram)
- ☒ (B) Trình cộng hành (Coroutine)
- (C) Gọi trở về đơn giản (Simple Call - Return)
- (D) Gọi đệ qui (Recursive call)
- (E) Biến cố - Xử lý biến cố (Exception)

29. Giả sử chương trình có một lỗi văn phạm nằm sau lệnh **print**. Khi em nhấn nút Run (để dịch và chạy chương trình) trên trình soạn thảo, kết quả được in ra bởi lệnh **print** trên trước khi lỗi văn phạm được báo. Hỏi chế độ dịch của trình soạn thảo là gì?

- (A) trình liên kết (link editor)
- ☒ (B) trình biên dịch động(just-in-time compiler)
- (C) trình biên dịch (compiler)
- ☒ (D) trình thông dịch(interpreter)
- (E) trình hợp ngữ (assembler)

30. Cho giá trị ban đầu của biến **c** là 3, cho biết những giá trị **có thể có** của biến **a** sau khi thực hiện phép gán sau trên ngôn ngữ lập trình C?

`a = c * (c = 5);`

- (A) 15, 25, 9
- ☒ (B) 15, 25
- (C) 15
- (D) 25
- (E) 9

31. Đoạn code viết bằng Scala nào dưới đây có thể dùng như một ví dụ minh họa cho khái niệm Hàm Currying

- (A) `def add1(n:Int) = n + 1`
- (B) `List(1,2,3).filter(_ > 1)`
- (C) `List(1,2,3).foldLeft(0)(_ + _)`
- ☒ (D) `def add(n:Int) = (x:Int) => n + x; val add2 = add(2)`
- ☒ (E) `def add(n:Int)(x:Int) = n + x; val add2 = add (2) _`

Đoạn mã sau được dùng trong các câu 32– 33:

```
int *p = new int;
void foo(int * r) {
    delete r;
}
foo(p); //1
```



*p = 2; //2

32. Hiện tượng gì xảy ra khi p được truyền cho r ở phát biểu //1 trong đoạn mã trên:

- (A) Con trỏ chưa khai báo (undeclared pointer) (B) Rác (garbage)
 (C) Bí danh (alias) (D) Khai báo trùng tên (redeclared)
 (E) Tham chiếu treo (dangling reference)

33. Hiện tượng gì xảy ra khi thực thi phát biểu //2 trong đoạn mã trên:

- (A) Tham chiếu treo (B) Con trỏ chưa khai báo (C) Bí danh
 (D) Đa hình (polymorphism) (E) Rác

34. Khi cần khai thác đặc tính của kiểu union (các thành phần loại trừ lẫn nhau), làm thế nào để hiện thực trên những ngôn ngữ lập trình hướng đối tượng như Java (không có kiểu union)? Cụ thể làm thế nào để hiện thực biến z kiểu union sau trên những ngôn ngữ này?

```
union {
    string loikhen;
    string loiche;
} z;
```

- (A) Định nghĩa lớp cha A và hai lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A
 (B) Định nghĩa lớp trừu tượng (abstract class) A và 2 lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A
 (C) Định nghĩa một lớp A có 2 thuộc tính loikhen và loiche; và z có kiểu A
 (D) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu A
 (E) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu B

35. Trình định thời (scheduled subprograms) thường được dùng trong

- (A) Lập trình song song (Parallel Programming)
 (B) Lập trình thời gian thực (Real-time Programming)
 (C) Lập trình hướng đối tượng (Object-Oriented Programming)
 (D) Lập trình hàm (Functional Programming)
 (E) Lập trình hướng sự kiện (Event-driven Programming)

36. Cho biết mã Jasmin của phát biểu gán sau:

$a[i] = a[i] + 2$

với i kiểu nguyên có chỉ số là 3 và a là kiểu dãy nguyên có chỉ số là 1.

- | | | | | | | | | | |
|-----|---|-----|--|-----|--|-----|--|-----|--|
| (A) | iastore
iadd
iaload
aload_1
iload_3
dup
iconst_2
... | (B) | aload_1
iload_3
iastore
aload_1
iload_3
iconst_2
iadd
... | (C) | aload_1
iload_3
iaload
iconst_2
iadd
iastore
... | (D) | aload_1
iload_3
aload_1
iload_3
iaload
iconst_2
iadd
iastore
... | (E) | aload_1
dup
iload_3
dup
iaload
iconst_2
iadd
iastore
... |
|-----|---|-----|--|-----|--|-----|--|-----|--|



37. Chọn phát biểu ĐÚNG trong các phát biểu về dãy (array) sau:

- (A) Trên C, kích thước của dãy có thể tăng thêm sau khi dãy đã được tạo ra
- ☒ (B) Truy xuất một phần tử bên ngoài một dãy Java không được phép, sẽ lập tức gây ra lỗi
- (C) Truy xuất một phần tử bên ngoài một dãy C không được phép, sẽ lập tức gây ra lỗi
- (D) Trên C và C++, kích thước của một dãy được lưu trữ để kiểm tra động
- (E) Trên Java, kích thước của một dãy KHÔNG được lưu trữ để kiểm tra động

Đoạn code sau dùng cho các câu 38–40

```
int p;  
int* foo(int x) {  
    static int q;  
    int *s = new int;  
    switch (x) {  
        case 1: return &p;  
        case 2: return &q;  
        case 3: return &x;  
        case 4: return s;  
        default: return foo(x-1);  
    }  
}
```

38. Phát biểu nào sẽ gây ra lỗi tham chiếu treo (dangling reference) khi thực thi

- (A) return s
- (B) return &p
- (C) return &q
- ☒ (D) return &x
- (E) Không phát biểu nào gây ra lỗi trên khi thực thi

39. Trong đoạn mã trên, đối tượng dữ liệu nào có thể trở thành rác (garbage)

- ☒ (A) trỏ đến bởi s
- (B) p
- (C) q
- (D) x
- (E) Không có đối tượng nào có thể trở thành rác

40. Khi hàm **foo** được gọi đệ quy (recursive) thì các bản hoạt động của **foo** dùng chung những đối tượng dữ liệu nào?

- (A) q và s
- (B) x, q và s
- (C) p, q, x và s
- (D) p
- ☒ (E) p và q

41. Gọi append là hàm dùng để nối 2 danh sách thành 1 danh sách. Ví dụ `append([1,2,3],[4,5,6])` sẽ có kết quả là `[1,2,3,4,5,6]`. Hãy hiện thực hàm `append(a:List[Int],b:List[Int])` dùng hàm bậc cao (high-order function)?

- (A) `a.map(b)((x,y)=>x::y)`
- (B) `b.foldRight(a)((x,y)=>x::y)`
- (C) `a.foldLeft(b)((x,y)=>x::y)`
- ☒ (D) `a.foldRight(b)((x,y)=>x::y)`
- (E) `b.foldLeft(a)((x,y)=>x::y)`

42. Một biến trên ngôn ngữ Javascript có thể nhận những giá trị thuộc các kiểu khác nhau. Ví dụ:

```
var x = "def";  
x = 1;
```

Ràng buộc giữa một biến và kiểu của nó trên ngôn ngữ Javascript xảy ra trong thời gian

- (A) Hiện thực (implementation)
- (B) Khởi động (loading)
- ☒ (C) Chạy (running)
- (D) Lập trình (programming)
- (E) Dịch (compiling)



Đoạn code sau, được viết trên ngôn ngữ dùng **qui tắc tầm vực tĩnh (static-scope rule)**, áp dụng cho các câu 43–46

```
procedure main(){
  var a, b, c:integer; //1
  procedure sub1(a: integer) { //2
    procedure sub3();
    procedure sub2() {
      var a, c : real; //3
      sub3();
    }
    procedure sub3() {
      a // use a
    }
    sub2();
  }
  sub1(3);
}
```

43. Môi trường tham khảo tĩnh (static referencing environment) của thủ tục **sub3** KHÔNG chứa

- (A) sub2 ☒ (B) a ở //1 (C) a ở //2 (D) sub1 (E) sub3

44. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong **sub3** ứng với khai báo

- (A) Báo lỗi a chưa khai báo (B) a trong main (//1) ☒ (C) a trong sub1 (//2)
(D) a trong sub2 (//3) (E) a trong sub3

45. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong sub3 **khi thực thi** ứng với kết hợp của **a** trong bản hoạt động của

- (A) main (B) Báo lỗi không tìm thấy a ☒ (C) sub1 (D) sub2
(E) sub3

46. Nếu đoạn code trên được viết trên ngôn ngữ dùng qui tắc tầm vực động (dynamic-scope rule), và giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến a trong sub3 **khi thực thi** ứng với kết hợp của a trong bản hoạt động của

- (A) Báo lỗi không tìm thấy a (B) sub3 (C) main (D) sub1
☒ (E) sub2

47. Cho $X = \{a, b\}$. Chọn biểu thức chính quy (regular expression) mô tả ngôn ngữ chứa bất kỳ chuỗi nào được tạo bởi các ký tự trên tập X nhưng KHÔNG chứa chuỗi có 2 ký tự **a** liên tiếp.

- ☒ (A) $b^*a?(bb^*a)^*b^*$ (B) $a|(abb^*)^*$ (C) $(b^*ab^*)^*$
(D) $b^*(abb^*)^*$ (E) $b^*ab^*ab^*$

II. Phần câu hỏi tự luận: (dành cho tất cả sinh viên) (2 điểm)



- 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

Lời giải. $(- (+ (* a b c) d e) f)$

Nếu viết đúng biểu thức dạng tiền tố Cambridge Polish: 1

Nếu viết đúng và có số () ít nhất: 1

III. Phần bài tập lớn: (dành cho tất cả sinh viên)

51. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện kiểm tra kiểu. Hàm `visitBlock` sẽ thực hiện chức năng xây dựng bảng danh hiệu (symbol table) và truyền bảng này khi visit các phát biểu trong khối. Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm. Nhắc lại AST của Block được khai báo như sau:
- ```
case class Block(val decl:List[Decl],val stmts:List[Stmt]) extends Stmt
```
52. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện sinh mã cho một phát biểu If. Nhắc lại AST của If được khai báo như sau:
- ```
case class If(val expr:Expr, val thenStmt:Stmt, val elseStmt:Option[Stmt]) extends Stmt
```
- Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm.

IV. Phần KSTN: (dành cho sinh viên KSTN)

53. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện kiểm tra lỗi FunctionNotReTurn? Yêu cầu giải thích ý tưởng thực hiện thể hiện qua mã của hàm?
54. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện sinh mã cho một khối (block)? Yêu cầu giải thích ý tưởng thực hiện sinh mã cho các khai báo array trong một khối?

Question Distribution

Content	Level 1	Level 2	Level 3	Level 4
Introduction		1		
Lexical	1	4	5	
Syntax	2	4	7	
AST		2		1
Total	3	11	12	1

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
BỘ MÔN KHOA HỌC MÁY TÍNH

Họ và tên:.....
MSSV:.....

Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình

Thời gian: 120 phút

Ngày thi: 23-12-2015

☐ Sinh viên được phép sử dụng tài liệu

☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1153

- Sinh viên phải ghi tên và mã số sinh viên trên đề thi (**trang đầu và trang cuối**), giấy làm bài trắc nghiệm (giấy đỏ) và giấy làm bài. Sinh viên phải tô phần mã đề thi và mã số sinh viên trên giấy làm bài trắc nghiệm. Khi nộp bài, sinh viên phải nộp cả **đề thi, giấy làm bài trắc nghiệm và giấy làm bài**.
- Phần trắc nghiệm sẽ được chấm TỰ ĐỘNG trên giấy làm bài trắc nghiệm. Do đó, phần trắc nghiệm nếu làm trên đề thi sẽ KHÔNG được chấm.
- Đối với các câu hỏi phần trắc nghiệm, sinh viên chỉ chọn MỘT phương án đúng nhất.
- Đối với phần câu hỏi tự luận (phần II), sinh viên làm ngay trên đề thi, ở phần dành riêng ngay dưới mỗi câu hỏi.
- Đối với câu hỏi bài tập lớn (phần III và IV), sinh viên làm trên giấy làm bài.
- Sinh viên lớp đại trà làm 3 phần (I, II và III). Sinh viên lớp tài năng làm cả 4 phần (I, II, III và IV). Chỉ có phần I và II được dùng để tính điểm cuối kỳ, các phần III và IV để tính điểm bài tập lớn 2 và 3.

I. Phần câu hỏi trắc nghiệm:(8 điểm)

1. Liệt kê theo thứ tự chiều dài tăng dần ít nhất 5 chuỗi ngắn nhất (nếu có nhiều hơn 5 chuỗi) của ngôn ngữ được mô tả bởi biểu thức chính quy (regular expression) sau:

$ab^*(a|b)?a$

- (A) aa, aba, aaa, abaa, aaba (B) ϵ , a, aa, aba, abaa (C) aa, aba, aaa, abaa, aaaa
(D) aa, aba, aaa, abaa, abba (E) a, aa, aba, aaa, aaba

2. Gọi append là hàm dùng để nối 2 danh sách thành 1 danh sách. Ví dụ `append([1,2,3],[4,5,6])` sẽ có kết quả là `[1,2,3,4,5,6]`. Hãy hiện thực hàm `append(a:List[Int],b:List[Int])` dùng hàm bậc cao (high-order function)?

- (A) `a.map(b)((x,y)=>x::y)` (B) `a.foldLeft(b)((x,y)=>x::y)`
(C) `b.foldRight(a)((x,y)=>x::y)` (D) `a.foldRight(b)((x,y)=>x::y)`
(E) `b.foldLeft(a)((x,y)=>x::y)`

3. Đoạn code viết bằng Scala nào dưới đây có thể dùng như một ví dụ minh họa cho khái niệm Hàm Currying

- (A) `def add1(n:Int) = n + 1`
(B) `List(1,2,3).foldLeft(0)(_ + _)`
(C) `List(1,2,3).filter(_ > 1)`
(D) `def add(n:Int) = (x:Int) => n + x; val add2 = add(2)`
(E) `def add(n:Int)(x:Int) = n + x; val add2 = add(2) _`

4. Cho giá trị ban đầu của biến **c** là 3, cho biết những giá trị **có thể có** của biến **a** sau khi thực hiện phép gán sau trên ngôn ngữ lập trình C?

$a = c * (c = 5);$

- (A) 15, 25, 9 (B) 15 (C) 15, 25 (D) 25 (E) 9

Cho văn phạm của một biểu thức sau dùng cho các câu 5–7:

$\text{exp} \rightarrow \text{term} '=' \text{exp} \mid \text{term}$

$\text{term} \rightarrow \text{term} '+' \text{fact} \mid \text{term} '>' \text{fact} \mid \text{fact}$

$\text{fact} \rightarrow \text{ope} '*' \text{ope} \mid \text{ope}$

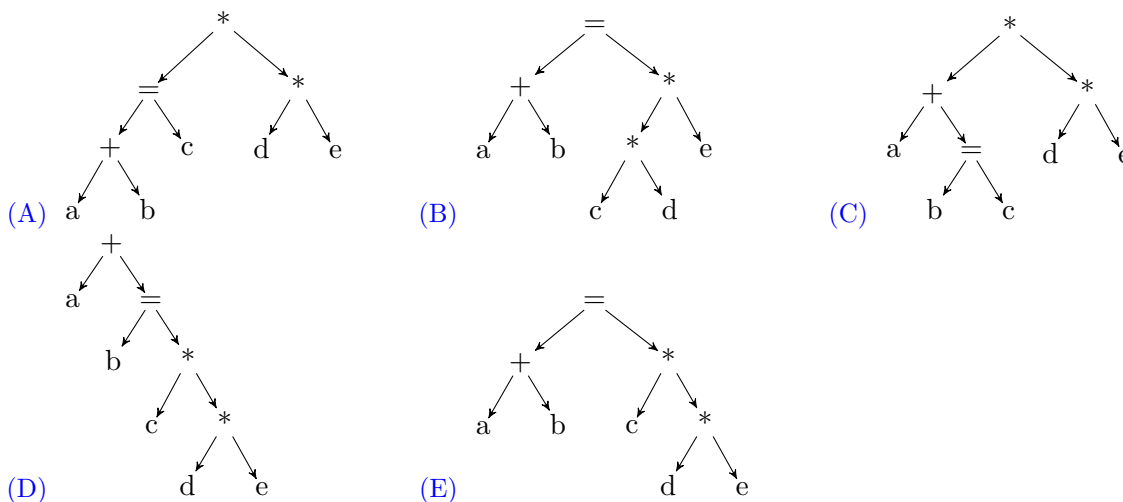
$\text{ope} \rightarrow '(' \text{exp} ')' \mid \text{ID}$

với ID là một danh hiệu.

5. Chọn biểu thức được viết đúng văn phạm, có số lượng dấu '(' và ')' ít nhất mà tương đương với biểu thức sau: $(a + (b = ((c * d) * e))) > f$

- (A) $(a + (b = (c * d) * e)) > f$ (B) $(a + (b = ((c * d) * e))) > f$
 (C) $a + (b = c * d * e) > f$ (D) $a + b = c * d * e > f$
 (E) $a + (b = (c * d) * e) > f$

6. Vẽ AST cho biểu thức sau được viết dựa vào văn phạm trên: $a + b = c * (d * e)$



7. Cho cấu trúc AST được khai báo trên Scala như sau:

trait Exp

case class Bin(op:String,e1:Exp,e2:Exp) extends Exp

case class Id(i:String) extends Exp

Chọn AST cho biểu thức sau: $a + b > c = d * e = f$

- (A) $\text{Bin}("=", \text{Bin}(">", \text{Bin}("+", \text{Id}("a"), \text{Id}("b")), \text{Id}("c")), \text{Bin}("*", \text{Id}("d"), \text{Bin}("=", \text{Id}("e"), \text{Id}("f"))))$
 (B) $\text{Bin}("=", \text{Bin}("=", \text{Bin}(">", \text{Bin}("+", \text{Id}("a"), \text{Id}("b")), \text{Id}("c")), \text{Bin}("*", \text{Id}("d"), \text{Id}("e"))), \text{Id}("f"))$
 (C) $\text{Bin}("=", \text{Bin}("=", \text{Bin}("+", \text{Id}("a"), \text{Bin}(">", \text{Id}("b"), \text{Id}("c"))), \text{Bin}("*", \text{Id}("d"), \text{Id}("e"))), \text{Id}("f"))$
 (D) $\text{Bin}("=", \text{Bin}(">", \text{Bin}("+", \text{Id}("a"), \text{Id}("b")), \text{Id}("c")), \text{Bin}("=", \text{Bin}("*", \text{Id}("d"), \text{Id}("e")), \text{Id}("f"))$
 (E) $\text{Bin}("=", \text{Bin}("+", \text{Id}("a"), \text{Bin}(">", \text{Id}("b"), \text{Id}("c"))), \text{Bin}("=", \text{Bin}("*", \text{Id}("d"), \text{Id}("e")), \text{Id}("f"))$



8. Cho lớp(class) A là lớp cha (superclass) của lớp B và B là lớp cha của hai lớp C và D. Trong lớp A, có khai báo phương thức thực thể (instance method) **foo** và phương thức này bị ghi đè(override) trên tất cả các lớp B, C và D. Cho biến x được khai báo như sau:
B x;

Qui ước viết $func_T$ là phương thức func được khai báo trong lớp T, cho biết tập của phương thức đích của cuộc gọi sau: **x.foo();**

- (A) {foo_C,foo_D} (B) {foo_A,foo_B,foo_C,foo_D} (C) {foo_B,foo_C,foo_D}
(D) {foo_B} (E) {foo_A,foo_B}

9. Trình định thời (scheduled subprograms) thường được dùng trong

- (A) Lập trình song song (Parallel Programming)
(B) Lập trình hướng đối tượng (Object-Oriented Programming)
(C) Lập trình thời gian thực (Real-time Programming)
(D) Lập trình hàm (Functional Programming)
(E) Lập trình hướng sự kiện (Event-driven Programming)

Đoạn code sau, được viết trên ngôn ngữ dùng **qui tắc tầm vực tĩnh (static-scope rule)**, áp dụng cho các câu 10–13

```
procedure main(){
  var a, b, c:integer; //1
  procedure sub1(a: integer) { //2
    procedure sub3();
    procedure sub2() {
      var a, c : real;//3
      sub3();
    }
    procedure sub3() {
      a // use a
    }
    sub2();
  }
  sub1(3);
}
```

10. Môi trường tham khảo tĩnh (static referencing environment) của thủ tục **sub3** KHÔNG chứa

- (A) a ở //1 (B) sub2 (C) a ở //2 (D) sub1 (E) sub3

11. Giả sử chuỗi gọi là main → sub1 → sub2 → sub3, tham khảo đến **a** trong **sub3** ứng với khai báo

- (A) Báo lỗi a chưa khai báo (B) a trong sub1 (//2) (C) a trong main (//1)
(D) a trong sub2 (//3) (E) a trong sub3

12. Giả sử chuỗi gọi là main → sub1 → sub2 → sub3, tham khảo đến **a** trong sub3 **khi thực thi** ứng với kết hợp của **a** trong bản hoạt động của

- (A) main (B) sub1 (C) Báo lỗi không tìm thấy a (D) sub2
(E) sub3



13. Nếu đoạn code trên được viết trên ngôn ngữ dùng qui tắc tầm vực động (dynamic-scope rule), và giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến a trong sub3 **khi thực thi** ứng với kết hợp của a trong bản hoạt động của

(A) Báo lỗi không tìm thấy a (B) main (C) sub3 (D) sub1
(E) sub2

14. Kiểu *union* của ngôn ngữ C++ cho phép ghi vào ở thành phần này nhưng đọc lên ở thành phần khác như trong ví dụ sau:

```
union {  
    int songuyen;  
    char kytu[2];  
} x;  
x.songuyen = 12;  
cout << x.kytu[0] << " " << x.kytu[1] << endl;
```

Đặc tính này của ngôn ngữ C++ gây ảnh hưởng XẤU NHẤT đối với

(A) tính trực giao (orthogonality) (B) tính đơn giản (simplicity)
(C) chi phí (cost) (D) tính dễ viết (writability)
(E) tính tin cậy (reliability)

15. Khi cần khai thác đặc tính của kiểu union (các thành phần loại trừ lẫn nhau), làm thế nào để hiện thực trên những ngôn ngữ lập trình hướng đối tượng như Java (không có kiểu union)? Cụ thể làm thế nào để hiện thực biến z kiểu union sau trên những ngôn ngữ này?

```
union {  
    string loikhen;  
    string loiche;  
} z;
```

(A) Định nghĩa lớp cha A và hai lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A
(B) Định nghĩa một lớp A có 2 thuộc tính loikhen và loiche; và z có kiểu A
(C) Định nghĩa lớp trừu tượng (abstract class) A và 2 lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A
(D) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu A
(E) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu B

16. Để thực hiện tìm kiếm một chuỗi (String) trên nhiều loại danh sách khác nhau, hãy chọn giải pháp thích hợp nhất để điền vào chỗ trống trong định nghĩa hàm sau:

```
def lookup[T](x:String, lst:List[T], f:T=>String):Option[T] =
  lst match {
    case List() => None
    case h::t =>_____
  }
```

- (A) if (h == x) Some(h) else lookup(x,t,f)
- (B) if (f(h) == x) Some(h) else lookup(x,t,f)
- (C) if (h == f(x)) Some(f(h)) else lookup(x,t,f)
- (D) if (h == f(x)) Some(h) else lookup(x,t,f)
- (E) if (f(h) == x) Some(f(h)) else lookup(x,t,f)

17. Cơ chế gọi chương trình con nào mà điều khiển có thể chuyển vào một vị trí ở giữa chương trình con được gọi thay vì chuyển vào đầu chương trình con được gọi?

- (A) Trình định thời (Scheduled Subprogram)
- (B) Gọi trở về đơn giản (Simple Call - Return)
- (C) Trình cộng hành (Coroutine)
- (D) Gọi đệ qui (Recursive call)
- (E) Biến cố - Xử lý biến cố (Exception)

Đoạn mã sau được dùng cho các câu 18-23.

Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:

```
int A[5] = {1,3,5,7,9}; // index of A starts from 0
int j = 0;
int n = 5;
int sumAndIncrease(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 = sumAndIncrease(A[j], j);
    printf("a = %i\n", s); //1
    printf(" %i %i %i %i %i\n", A[0], A[1], A[2], A[3], A[4]); //2
}
```

18. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5
- (B) 25
- (C) 10
- (D) 15
- (E) Khác

19. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9
- (B) 5 3 5 7 9
- (C) 2 4 6 8 10
- (D) 6 3 5 7 9
- (E) Khác



20. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 (B) 25 (C) 10 (D) 15 (E) Khác

21. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 5 3 5 7 9 (C) 2 4 6 8 10 (D) 6 3 5 7 9 (E) Khác

22. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 (B) 25 (C) 15 (D) 20 (E) Khác

23. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 5 3 5 7 9 (C) 2 4 6 8 10 (D) 6 3 5 7 9 (E) Khác

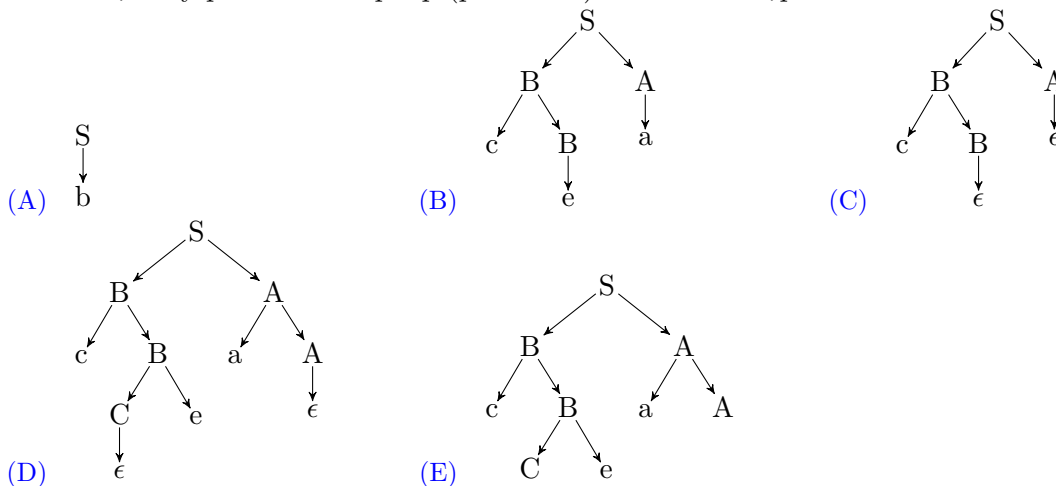
Cho văn phạm sau dùng cho các câu 24-25:

S	→	b		B A
A	→	a A		ε
B	→	C e		c B
C	→	d C		ε

24. Chuỗi nào KHÔNG thuộc ngôn ngữ được mô tả bởi văn phạm trên?

- (A) e (B) b (C) cccaaa (D) eaaa (E) cccddddeaa

25. Chọn cây phân tích cú pháp (parse tree) cho chuỗi nhập: cea



26. Viết lại biểu thức trung tố (infix) sau sang dạng hậu tố (postfix) Polish? Giả sử là tất cả các phép toán đều có 2 toán hạng và có độ ưu tiên và tính kết hợp như trong ngôn ngữ lập trình C.

$a + b * c - d * e + f$

- (A) $a + b c * d e * - f +$ (B) $a b + c * d - e * f +$ (C) $a b c * + d e * - f +$
 (D) $a b c * + d - e f * +$ (E) $a b c * + d e * + f -$

Đoạn code sau dùng cho các câu 27-29

```
int p;
int* foo(int x) {
```

```
static int q;  
int *s = new int;  
switch (x) {  
    case 1: return &p;  
    case 2: return &q;  
    case 3: return &x;  
    case 4: return s;  
    default: return foo(x-1);  
}  
}
```

27. Phát biểu nào sẽ gây ra lỗi tham chiếu treo (dangling reference) khi thực thi
(A) return &p (B) return s (C) return &q (D) return &x
(E) Không phát biểu nào gây ra lỗi trên khi thực thi
28. Trong đoạn mã trên, đối tượng dữ liệu nào có thể trở thành rác (garbage)
(A) p (B) trở đến bởi s (C) q (D) x
(E) Không có đối tượng nào có thể trở thành rác
29. Khi hàm **foo** được gọi đệ quy (recursive) thì các bản hoạt động của **foo** dùng chung những đối tượng dữ liệu nào?
(A) q và s (B) p, q, x và s (C) x, q và s (D) p (E) p và q
30. Một biến trên ngôn ngữ Javascript có thể nhận những giá trị thuộc các kiểu khác nhau.
Ví dụ:
var x = "def";
x = 1;
Ràng buộc giữa một biến và kiểu của nó trên ngôn ngữ Javascript xảy ra trong thời gian
(A) Hiện thực (implementation) (B) Chạy (running)
(C) Khởi động (loading) (D) Lập trình (programming)
(E) Dịch (compiling)
31. Cơ chế gọi chương trình nào là cơ chế gọi chương trình con cơ bản của Lập trình hướng sự kiện (Event-driven Programming)
(A) Biến cố - Xử lý biến cố (Exception)
(B) Gọi trở về đơn giản (Simple Call - Return)
(C) Trình định thời (Scheduled Subprogram)
(D) Gọi đệ qui (Recursive call)
(E) Trình cộng hành (Coroutine)
32. Chọn phát biểu ĐÚNG trong các phát biểu về dãy (array) sau:
(A) Trên C, kích thước của dãy có thể tăng thêm sau khi dãy đã được tạo ra
(B) Truy xuất một phần tử bên ngoài một dãy C không được phép, sẽ lập tức gây ra lỗi
(C) Truy xuất một phần tử bên ngoài một dãy Java không được phép, sẽ lập tức gây ra lỗi
(D) Trên C và C++, kích thước của một dãy được lưu trữ để kiểm tra động
(E) Trên Java, kích thước của một dãy KHÔNG được lưu trữ để kiểm tra động

Phần trình bày sau dùng trong các câu hỏi 33– 34:

Cho mã giả của phát biểu *for var = expr1 to expr2 do body* như sau:



```

start:  var = expr1
        etemp = expr2
loop:   if (var >= etemp) goto out
        body
label1: var++
label2: goto loop
out:

```

33. Nếu trong thân *body* có thực thi lệnh *continue* thì điều khiển sẽ chuyển đến thực thi phát biểu có nhãn là:

- (A) out (B) start (C) label2 (D) loop (E) label1

34. Với hiện thực của phát biểu *for* như trên, phát biểu nào đặt trong thân phát biểu **for** dưới đây sẽ làm cho phát biểu này bị lặp mãi mãi?

```

n = 10; s = 0;
for i = s to n do -----

```

- (A) *i = i - 1;* (B) *s = s - 1;* (C) *n = n + 1;*
 (D) câu A và C đúng
 (E) Không phát biểu nào trong các câu A, B và C làm phát biểu **for** lặp mãi mãi

Phần hướng dẫn này áp dụng cho các câu 35–39

Hãy viết thêm các đoạn mã cần thiết vào phương thức dưới đây để thực hiện sinh mã cho phát biểu *repeat*. Nhắc lại phát biểu *repeat* kết thúc quá trình lặp khi điều kiện của phát biểu trở thành **true**. AST của phát biểu **repeat** được định nghĩa như sau:

```
case class Repeat(val stmts: List[Stmt], val exp: Expr) extends Stmt
```

```

override def visitRepeat(ast: Repeat, o: Context) = {
  val ctxt = o.asInstanceOf[SubBody]
  ctxt.frame.enterLoop();
  val labelStart = ctxt.frame.getNewLabel()
  val labelBreak = ctxt.frame.getBreakLabel()
  val labelCont = ctxt.frame.getContinueLabel()
  val str1 = //1
  // sinh mã cho từng phát biểu trong thân của repeat
  val str2 = //2
  val str3 = //3
  // sinh mã cho biểu thức điều kiện
  val str4 = visit(ast.exp, ...)
  val str5 = //4
  val str6 = //5
  frame().exitLoop();
  str1 + str2 + str3 + str4 + str5 + str6
}

```

35. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //1 trong đoạn mã trên

- (A) "" (B) *ctxt.emit.emitLABEL(labelStart)*
 (C) *ctxt.emit.emitGOTO(labelStart)* (D) *ctxt.emit.emitLABEL(labelBreak)*
 (E) *ctxt.emit.emitLABEL(labelCont)*

36. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //2 trong đoạn mã trên

- (A) `visit(ast.stmts,o)`
- (B) `ast.stmts.map(x=>visit(x,o))`
- (C) `ast.stmts.foldLeft(")((x,y)=>y + visit(x,o))`
- (D) `ast.stmts.filter(x=>visit(x,o))`
- (E) `ast.stmts.foldLeft(")((x,y)=>x + visit(y,o))`

37. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //3 trong đoạn mã trên

- (A) `" "`
- (B) `ctxt.emit.emitIFFALSE(labelStart)`
- (C) `ctxt.emit.emitGOTO(labelStart)`
- (D) `ctxt.emit.emitLABEL(labelBreak)`
- (E) `ctxt.emit.emitLABEL(labelCont)`

38. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //4 trong đoạn mã trên

- (A) `ctxt.emit.emitGOTO(labelBreak)`
- (B) `ctxt.emit.emitIFFALSE(labelStart)`
- (C) `ctxt.emit.emitGOTO(labelStart)`
- (D) `ctxt.emit.emitLABEL(labelBreak)`
- (E) `ctxt.emit.emitLABEL(labelCont)`

39. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //5 trong đoạn mã trên

- (A) `ctxt.emit.emitGOTO(labelBreak))`
- (B) `ctxt.emit.emitIFFALSE(labelStart))`
- (C) `ctxt.emit.emitGOTO(labelStart))`
- (D) `ctxt.emit.emitLABEL(labelBreak))`
- (E) `ctxt.emit.emitLABEL(labelCont))`

40. Cho khai báo sau được viết trên ngôn ngữ lập trình Ada:

```
type Shape is ( Circle , Triangle , Rectangle );  
type Colors is ( Red, Green, Blue );  
type Figure (Form: Shape) is record  
    Filled: Boolean;  
    Color: Colors;  
    case Form is  
        when Circle => Diameter: Float;  
        when Triangle =>  
            Leftside, Rightside: Integer;  
            Angle: Float;  
        when Rectangle => Side1, Side2: Integer;  
    end case;  
end record;
```

Cho kích thước của các kiểu *Integer*, *Float*, *Boolean* và *Enumeration* lần lượt là 2, 4, 1, và 2, và kích thước cho phần mô tả kiểu (type description) là 0. Cho biết kích thước của một đối tượng kiểu **Figure**?

- (A) 21
- (B) 15
- (C) 13
- (D) 23
- (E) Khác

41. Cho biết kích thước của đối tượng dữ liệu **x** được khai báo như sau:
`x: set of 1..16`

- (A) 4 bytes
- (B) 2 bytes
- (C) 16 bytes
- (D) 4 bits
- (E) 1 byte

42. Cho $X = \{a, b\}$. Chọn biểu thức chính quy (regular expression) mô tả ngôn ngữ chứa bất kỳ chuỗi nào được tạo bởi các ký tự trên tập X nhưng KHÔNG chứa chuỗi có 2 ký tự a liên tiếp.

- (A) $a|(abb^*)^*$ (B) $b^*a?(bb^*a)^*b^*$ (C) $(b^*ab^*)^*$ (D) $b^*(abb^*)^*$ (E) $b^*ab^*ab^*$

43. Cho biết mã Jasmin của biểu thức viết bằng ngôn ngữ C như sau:

$a * b - 2$

với a và b là 2 biến nguyên có chỉ số lần lượt là 0 và 1.

- | | | | | | |
|-----|----------|--------------|-------------|-------------|--------------|
| | iload_0 | iload_0 | imul | iload_0 | iload_0 |
| | iload_1 | iload_1 | load_0 | imul | iload_1 |
| (A) | imul | (B) iconst_2 | (C) iload_1 | (D) iload_1 | (E) iconst_2 |
| | iconst_2 | imul | isub | isub | isub |
| | isub | isub | iconst_2 | iconst_2 | imul |
| | ... | ... | ... | ... | ... |

Đoạn mã sau được dùng trong các câu 44– 45:

```
int *p = new int;
void foo(int * r) {
    delete r;
}
foo(p); //1
*p = 2; //2
```

44. Hiện tượng gì xảy ra khi p được truyền cho r ở phát biểu //1 trong đoạn mã trên:

- (A) Con trỏ chưa khai báo (undeclared pointer) (B) Bí danh (alias)
 (C) Rác (garbage) (D) Khai báo trùng tên (redeclared)
 (E) Tham chiếu treo (dangling reference)

45. Hiện tượng gì xảy ra khi thực thi phát biểu //2 trong đoạn mã trên:

- (A) Tham chiếu treo (B) Bí danh (C) Con trỏ chưa khai báo
 (D) Đa hình (polymorphism) (E) Rác

46. Giả sử chương trình có một lỗi văn phạm nằm sau lệnh **print**. Khi em nhấn nút Run (để dịch và chạy chương trình) trên trình soạn thảo, kết quả được in ra bởi lệnh **print** trên trước khi lỗi văn phạm được báo. Hỏi chế độ dịch của trình soạn thảo là gì?

- (A) trình liên kết (link editor) (B) trình biên dịch (compiler)
 (C) trình biên dịch động(just-in-time compiler) (D) trình thông dịch(interpreter)
 (E) trình hợp ngữ (assembler)

47. Cho biết mã Jasmin của phát biểu gán sau:

$a[i] = a[i] + 2$

với i kiểu nguyên có chỉ số là 3 và a là kiểu dãy nguyên có chỉ số là 1.

- | | | | | | |
|-----|----------|--------------|-------------|------------|------------|
| | iastore | aload_1 | aload_1 | aload_1 | aload_1 |
| | iadd | iload_3 | iload_3 | iload_3 | dup |
| | iaload | iaload | iastore | aload_1 | iload_3 |
| (A) | aload_1 | (B) iconst_2 | (C) aload_1 | (D) iaload | (E) iaload |
| | iload_3 | iadd | iload_3 | iconst_2 | iconst_2 |
| | dup | iastore | iconst_2 | iadd | iadd |
| | iconst_2 | ... | iadd | iastore | iastore |
| | ... | ... | ... | ... | ... |



- 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

III. Phần bài tập lớn: (dành cho tất cả sinh viên)

51. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện kiểm tra kiểu. Hàm `visitBlock` sẽ thực hiện chức năng xây dựng bảng danh hiệu (symbol table) và truyền bảng này khi `visit` các phát biểu trong khối. Yêu cầu giải thích rõ về `Context` cụ thể được sử dụng trong hàm và ý tưởng mã của hàm. Nhắc lại AST của `Block` được khai báo như sau:

```
case class Block(val decl:List[Decl],val stmts:List[Stmt]) extends Stmt
```

52. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện sinh mã cho một phát biểu `If`. Nhắc lại AST của `If` được khai báo như sau:

```
case class If(val expr:Expr, val thenStmt:Stmt, val elseStmt:Option[Stmt]) extends Stmt
```

Yêu cầu giải thích rõ về `Context` cụ thể được sử dụng trong hàm và ý tưởng mã của hàm.

IV. Phần KSTN: (dành cho sinh viên KSTN)

53. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện kiểm tra lỗi `FunctionNotReTurn`? Yêu cầu giải thích ý tưởng thực hiện thể hiện qua mã của hàm?
54. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện sinh mã cho một khối (block)? Yêu cầu giải thích ý tưởng thực hiện sinh mã cho các khai báo array trong một khối?

HẾT

Họ và tên:.....	Điểm:
MSSV:.....	



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



KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH
BỘ MÔN KHOA HỌC MÁY TÍNH

ĐÁP ÁN cho Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình

Thời gian: 120 phút

☐ Sinh viên được phép sử dụng tài liệu

☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1153

I. Phần câu hỏi trắc nghiệm:(8 điểm))

1. Liệt kê theo thứ tự chiều dài tăng dần ít nhất 5 chuỗi ngắn nhất (nếu có nhiều hơn 5 chuỗi) của ngôn ngữ được mô tả bởi biểu thức chính quy (regular expression) sau:
 $ab^*(a|b)?a$

(A) aa, aba, aaa, abaa, aaba (B) ϵ , a, aa, aba, abaa (C) aa, aba, aaa, abaa, aaaa
(D) aa, aba, aaa, abaa, abba (E) a, aa, aba, aaa, aaba

2. Gọi append là hàm dùng để nối 2 danh sách thành 1 danh sách. Ví dụ `append([1,2,3],[4,5,6])` sẽ có kết quả là `[1,2,3,4,5,6]`. Hãy hiện thực hàm `append(a:List[Int],b:List[Int])` dùng hàm bậc cao (high-order function)?

(A) `a.map(b)((x,y)=>x::y)` (B) `a.foldLeft(b)((x,y)=>x::y)`
(C) `b.foldRight(a)((x,y)=>x::y)` (D) `a.foldRight(b)((x,y)=>x::y)`
(E) `b.foldLeft(a)((x,y)=>x::y)`

3. Đoạn code viết bằng Scala nào dưới đây có thể dùng như một ví dụ minh họa cho khái niệm Hàm Currying

(A) `def add1(n:Int) = n + 1`
(B) `List(1,2,3).foldLeft(0)(_ + _)`
(C) `List(1,2,3).filter(_ > 1)`
(D) `def add(n:Int) = (x:Int) => n + x; val add2 = add(2)`
(E) `def add(n:Int)(x:Int) = n + x; val add2 = add(2) _`

4. Cho giá trị ban đầu của biến **c** là 3, cho biết những giá trị **có thể có** của biến **a** sau khi thực hiện phép gán sau trên ngôn ngữ lập trình C?
`a = c * (c = 5);`

(A) 15, 25, 9 (B) 15 (C) 15, 25 (D) 25 (E) 9

Cho văn phạm của một biểu thức sau dùng cho các câu 5–7:

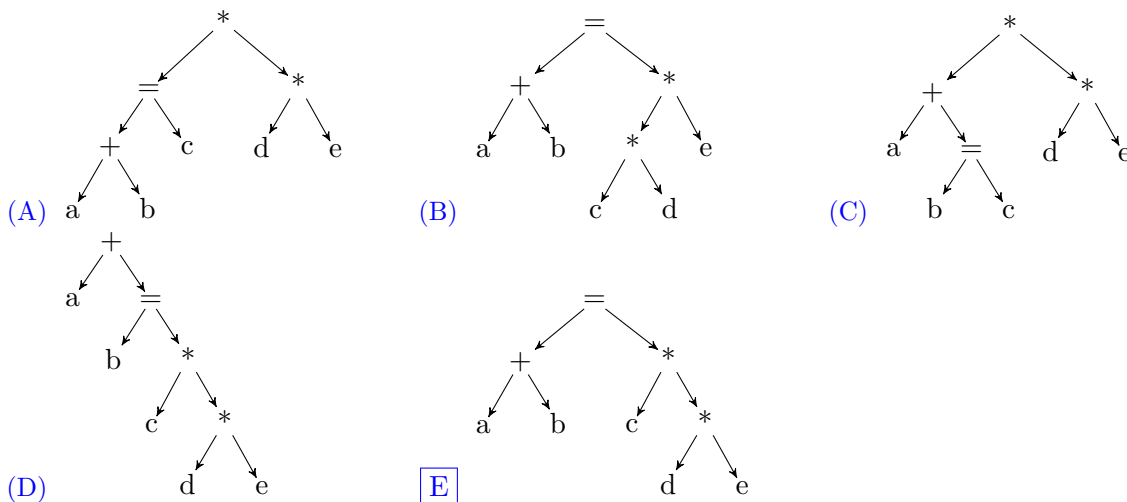
`exp` \rightarrow `term` '=' `exp` | `term`
`term` \rightarrow `term` '+' `fact` | `term` '>' `fact` | `fact`
`fact` \rightarrow `ope` '*' `ope` | `ope`
`ope` \rightarrow '(' `exp` ')' | ID

với ID là một danh hiệu.

5. Chọn biểu thức được viết đúng văn phạm, có số lượng dấu '(' và ')' ít nhất mà tương đương với biểu thức sau: $(a + (b = ((c * d) * e))) > f$

- (A) $(a + (b = (c * d) * e)) > f$ (B) $(a + (b = ((c * d) * e))) > f$
 (C) $a + (b = c * d * e) > f$ (D) $a + b = c * d * e > f$
 (E) $a + (b = (c * d) * e) > f$

6. Vẽ AST cho biểu thức sau được viết dựa vào văn phạm trên: $a + b = c * (d * e)$



7. Cho cấu trúc AST được khai báo trên Scala như sau:

```
trait Exp
case class Bin(op:String,e1:Exp,e2:Exp) extends Exp
case class Id(i:String) extends Exp
```

Chọn AST cho biểu thức sau: $a + b > c = d * e = f$

- (A) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Bin("=",Id("e"),Id("f"))))`
 (B) `Bin("=",Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Id("e"))),Id("f"))`
 (C) `Bin("=",Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("*",Id("d"),Id("e"))),Id("f"))`
 (D) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("=",Bin("*",Id("d"),Id("e")),Id("f"))`
 (E) `Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("=",Bin("*",Id("d"),Id("e")),Id("f"))`

8. Cho lớp(class) A là lớp cha (superclass) của lớp B và B là lớp cha của hai lớp C và D. Trong lớp A, có khai báo phương thức thực thể (instance method) **foo** và phương thức này bị ghi đè(override) trên tất cả các lớp B, C và D. Cho biến x được khai báo như sau: **B x;**

Qui ước viết $func_T$ là phương thức func được khai báo trong lớp T, cho biết tập của phương thức đích của cuộc gọi sau: **x.foo();**

- (A) $\{foo_C, foo_D\}$ (B) $\{foo_A, foo_B, foo_C, foo_D\}$ (C) $\{foo_B, foo_C, foo_D\}$
 (D) $\{foo_B\}$ (E) $\{foo_A, foo_B\}$



9. Trình định thời (scheduled subprograms) thường được dùng trong

- (A) Lập trình song song (Parallel Programming)
- (B) Lập trình hướng đối tượng (Object-Oriented Programming)
- ☒ (C) Lập trình thời gian thực (Real-time Programming)
- (D) Lập trình hàm (Functional Programming)
- (E) Lập trình hướng sự kiện (Event-driven Programming)

Đoạn code sau, được viết trên ngôn ngữ dùng **qui tắc tầm vực tĩnh (static-scope rule)**, áp dụng cho các câu 10–13

```
procedure main(){
  var a, b, c:integer; //1
  procedure sub1(a: integer) { //2
    procedure sub3();
    procedure sub2() {
      var a, c : real; //3
      sub3();
    }
    procedure sub3() {
      a // use a
    }
    sub2();
  }
  sub1(3);
}
```

10. Môi trường tham khảo tĩnh (static referencing environment) của thủ tục **sub3** KHÔNG chứa

- ☒ (A) a ở //1 (B) sub2 (C) a ở //2 (D) sub1 (E) sub3

11. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong **sub3** ứng với khai báo

- (A) Báo lỗi a chưa khai báo ☒ (B) a trong sub1 (//2) (C) a trong main (//1)
(D) a trong sub2 (//3) (E) a trong sub3

12. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong sub3 **khi thực thi** ứng với kết hợp của **a** trong bản hoạt động của

- (A) main ☒ (B) sub1 (C) Báo lỗi không tìm thấy a (D) sub2
(E) sub3

13. Nếu đoạn code trên được viết trên ngôn ngữ dùng qui tắc tầm vực động (dynamic-scope rule), và giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến a trong sub3 **khi thực thi** ứng với kết hợp của a trong bản hoạt động của

- (A) Báo lỗi không tìm thấy a (B) main (C) sub3 (D) sub1
☒ (E) sub2

14. Kiểu *union* của ngôn ngữ C++ cho phép ghi vào ở thành phần này nhưng đọc lên ở thành phần khác như trong ví dụ sau:

```
union {  
    int songuyen;  
    char kytu[2];  
} x;  
x.songuyen = 12;  
cout << x.kytu[0] << " " << x.kytu[1] << endl;
```

Đặc tính này của ngôn ngữ C++ gây ảnh hưởng XẤU NHẤT đối với

- (A) tính trực giao (orthogonality) (B) tính đơn giản (simplicity)
(C) chi phí (cost) (D) tính dễ viết (writability)
☒ (E) tính tin cậy (reliability)

15. Khi cần khai thác đặc tính của kiểu *union* (các thành phần loại trừ lẫn nhau), làm thế nào để hiện thực trên những ngôn ngữ lập trình hướng đối tượng như Java (không có kiểu *union*)? Cụ thể làm thế nào để hiện thực biến *z* kiểu *union* sau trên những ngôn ngữ này?

```
union {  
    string loikhen;  
    string loiche;  
} z;
```

- (A) Định nghĩa lớp cha A và hai lớp con B (có thuộc tính *loikhen*) và C (có thuộc tính *loiche*), và *z* có kiểu A
(B) Định nghĩa một lớp A có 2 thuộc tính *loikhen* và *loiche*; và *z* có kiểu A
☒ (C) Định nghĩa lớp trừu tượng (abstract class) A và 2 lớp con B (có thuộc tính *loikhen*) và C (có thuộc tính *loiche*), và *z* có kiểu A
(D) Định nghĩa lớp A có thuộc tính *loikhen* và lớp B (con của A) có thuộc tính *loiche*, và *z* có kiểu A
(E) Định nghĩa lớp A có thuộc tính *loikhen* và lớp B (con của A) có thuộc tính *loiche*, và *z* có kiểu B

16. Để thực hiện tìm kiếm một chuỗi (String) trên nhiều loại danh sách khác nhau, hãy chọn giải pháp thích hợp nhất để điền vào chỗ trống trong định nghĩa hàm sau:

```
def lookup[T](x:String, lst:List[T], f:T=>String):Option[T] =  
    lst match {  
        case List() => None  
        case h::t => _____  
    }
```

- (A) if (h == x) Some(h) else lookup(x,t,f)
☒ (B) if (f(h) == x) Some(h) else lookup(x,t,f)
(C) if (h == f(x)) Some(f(h)) else lookup(x,t,f)
(D) if (h == f(x)) Some(h) else lookup(x,t,f)
(E) if (f(h) == x) Some(f(h)) else lookup(x,t,f)

17. Cơ chế gọi chương trình con nào mà điều khiển có thể chuyển vào một vị trí ở giữa chương trình con được gọi thay vì chuyển vào đầu chương trình con được gọi?

- (A) Trình định thời (Scheduled Subprogram)
- (B) Gọi trở về đơn giản (Simple Call - Return)
- ☒ (C) Trình cộng hành (Coroutine)
- (D) Gọi đệ qui (Recursive call)
- (E) Biến cố - Xử lý biến cố (Exception)

Đoạn mã sau được dùng cho các câu 18–23.

Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:

```
int A[5] = {1,3,5,7,9}; // index of A starts from 0
int j = 0;
int n = 5;
int sumAndIncrease(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 = sumAndIncrease(A[j], j);
    printf("a = %i\n", s); //1
    printf(" %i %i %i %i %i\n", A[0], A[1], A[2], A[3], A[4]); //2
}
```

18. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- ☒ (A) 5 (B) 25 (C) 10 (D) 15 (E) Khác

19. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 5 3 5 7 9 (C) 2 4 6 8 10 ☒ (D) 6 3 5 7 9 (E) Khác

20. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 (B) 25 (C) 10 (D) 15 ☒ (E) Khác

21. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 5 3 5 7 9 ☒ (C) 2 4 6 8 10 (D) 6 3 5 7 9 (E) Khác

22. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 ☒ (B) 25 (C) 15 (D) 20 (E) Khác

23. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 5 3 5 7 9 **(C) 2 4 6 8 10** (D) 6 3 5 7 9 (E) Khác

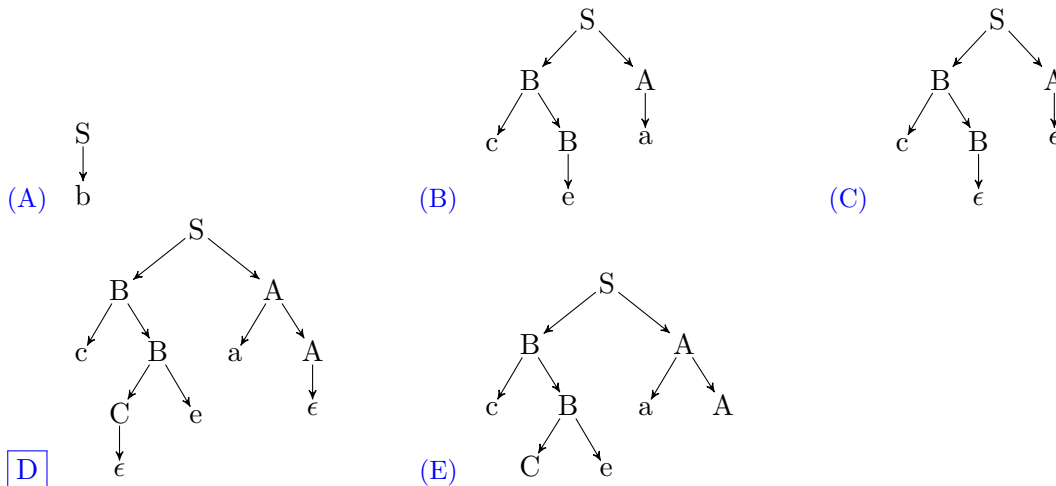
Cho văn phạm sau dùng cho các câu 24–25:

$$\begin{array}{lcl} S & \rightarrow & b \mid BA \\ A & \rightarrow & aA \mid \epsilon \\ B & \rightarrow & Ce \mid cB \\ C & \rightarrow & dC \mid \epsilon \end{array}$$

24. Chuỗi nào KHÔNG thuộc ngôn ngữ được mô tả bởi văn phạm trên?

- (A) e (B) b **(C) cccaaa** (D) eaaa (E) cccddddeaa

25. Chọn cây phân tích cú pháp (parse tree) cho chuỗi nhập: cea



26. Viết lại biểu thức trung tố (infix) sau sang dạng hậu tố (postfix) Polish? Giả sử là tất cả các phép toán đều có 2 toán hạng và có độ ưu tiên và tính kết hợp như trong ngôn ngữ lập trình C.

$a + b * c - d * e + f$

- (A) $a + b c * d e * - f +$ (B) $a b + c * d - e * f +$ **(C) $a b c * + d e * - f +$**
 (D) $a b c * + d - e f * +$ (E) $a b c * + d e * + f -$

Đoạn code sau dùng cho các câu 27–29

```
int p;
int* foo(int x) {
    static int q;
    int *s = new int;
    switch (x) {
        case 1: return &p;
        case 2: return &q;
        case 3: return &x;
        case 4: return s;
        default: return foo(x-1);
    }
}
```



27. Phát biểu nào sẽ gây ra lỗi tham chiếu treo (dangling reference) khi thực thi

- (A) `return &p` (B) `return s` (C) `return &q` ☒ (D) `return &x`
(E) Không phát biểu nào gây ra lỗi trên khi thực thi

28. Trong đoạn mã trên, đối tượng dữ liệu nào có thể trở thành rác (garbage)

- (A) `p` ☒ (B) trở đến bởi `s` (C) `q` (D) `x`
(E) Không có đối tượng nào có thể trở thành rác

29. Khi hàm **foo** được gọi đệ quy (recursive) thì các bản hoạt động của **foo** dùng chung những đối tượng dữ liệu nào?

- (A) `q` và `s` (B) `p`, `q`, `x` và `s` (C) `x`, `q` và `s` (D) `p` ☒ (E) `p` và `q`

30. Một biến trên ngôn ngữ Javascript có thể nhận những giá trị thuộc các kiểu khác nhau.
Ví dụ:

```
var x = "def";
```

```
x = 1;
```

Ràng buộc giữa một biến và kiểu của nó trên ngôn ngữ Javascript xảy ra trong thời gian

- (A) Hiện thực (implementation) ☒ (B) Chạy (running)
(C) Khởi động (loading) (D) Lập trình (programming)
(E) Dịch (compiling)

31. Cơ chế gọi chương trình nào là cơ chế gọi chương trình con cơ bản của Lập trình hướng sự kiện (Event-driven Programming)

- ☒ (A) Biến cố - Xử lý biến cố (Exception)
(B) Gọi trở về đơn giản (Simple Call - Return)
(C) Trình định thời (Scheduled Subprogram)
(D) Gọi đệ quy (Recursive call)
(E) Trình cộng hành (Coroutine)

32. Chọn phát biểu ĐÚNG trong các phát biểu về dãy (array) sau:

- (A) Trên C, kích thước của dãy có thể tăng thêm sau khi dãy đã được tạo ra
(B) Truy xuất một phần tử bên ngoài một dãy C không được phép, sẽ lập tức gây ra lỗi
☒ (C) Truy xuất một phần tử bên ngoài một dãy Java không được phép, sẽ lập tức gây ra lỗi
(D) Trên C và C++, kích thước của một dãy được lưu trữ để kiểm tra động
(E) Trên Java, kích thước của một dãy KHÔNG được lưu trữ để kiểm tra động

Phần trình bày sau dùng trong các câu hỏi 33– 34:

Cho mã giả của phát biểu *for var = expr1 to expr2 do body* như sau:

```
start:  var = expr1
        etemp = expr2
loop:   if (var >= etemp) goto out
        body
label1: var++
label2: goto loop
out:
```

33. Nếu trong thân *body* có thực thi lệnh *continue* thì điều khiển sẽ chuyển đến thực thi phát biểu có nhãn là:

- (A) out (B) start (C) label2 (D) loop **(E) label1**

34. Với hiện thực của phát biểu *for* như trên, phát biểu nào đặt trong thân phát biểu **for** dưới đây sẽ làm cho phát biểu này bị lặp mãi mãi?

```
n = 10; s = 0;
for i = s to n do -----
```

- (A)** i = i - 1; (B) s = s - 1; (C) n = n + 1;
(D) câu A và C đúng
(E) Không phát biểu nào trong các câu A, B và C làm phát biểu **for** lặp mãi mãi

Phần hướng dẫn này áp dụng cho các câu 35–39

Hãy viết thêm các đoạn mã cần thiết vào phương thức dưới đây để thực hiện sinh mã cho phát biểu *repeat*. Nhắc lại phát biểu *repeat* kết thúc quá trình lặp khi điều kiện của phát biểu trở thành **true**. AST của phát biểu **repeat** được định nghĩa như sau:

```
case class Repeat(val stmts: List[Stmt], val exp: Expr) extends Stmt
```

```
override def visitRepeat(ast: Repeat, o: Context) = {
  val ctxt = o.asInstanceOf[SubBody]
  ctxt.frame.enterLoop();
  val labelStart = ctxt.frame.getNewLabel()
  val labelBreak = ctxt.frame.getBreakLabel()
  val labelCont = ctxt.frame.getContinueLabel()
  val str1 = //1
  // sinh mã cho từng phát biểu trong thân của repeat
  val str2 = //2
  val str3 = //3
  // sinh mã cho biểu thức điều kiện
  val str4 = visit(ast.exp, ...)
  val str5 = //4
  val str6 = //5
  frame().exitLoop();
  str1 + str2 + str3 + str4 + str5 + str6
}
```

35. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //1 trong đoạn mã trên

- (A) "" **(B)** ctxt.emit.emitLABEL(labelStart)
(C) ctxt.emit.emitGOTO(labelStart) (D) ctxt.emit.emitLABEL(labelBreak)
(E) ctxt.emit.emitLABEL(labelCont)

36. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //2 trong đoạn mã trên

- (A) visit(ast.stmts, o)
(B) ast.stmts.map(x => visit(x, o))
(C) ast.stmts.foldLeft("")((x, y) => y + visit(x, o))
(D) ast.stmts.filter(x => visit(x, o))
(E) ast.stmts.foldLeft("")((x, y) => x + visit(y, o))



37. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //3 trong đoạn mã trên

- (A) "" (B) `ctxt.emit.emitIFFALSE(labelStart)`
(C) `ctxt.emit.emitGOTO(labelStart)` (D) `ctxt.emit.emitLABEL(labelBreak)`
(E) `ctxt.emit.emitLABEL(labelCont)`

38. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //4 trong đoạn mã trên

- (A) `ctxt.emit.emitGOTO(labelBreak)` (B) `ctxt.emit.emitIFFALSE(labelStart)`
(C) `ctxt.emit.emitGOTO(labelStart)` (D) `ctxt.emit.emitLABEL(labelBreak)`
(E) `ctxt.emit.emitLABEL(labelCont)`

39. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //5 trong đoạn mã trên

- (A) `ctxt.emit.emitGOTO(labelBreak)` (B) `ctxt.emit.emitIFFALSE(labelStart)`
(C) `ctxt.emit.emitGOTO(labelStart)` (D) `ctxt.emit.emitLABEL(labelBreak)`
(E) `ctxt.emit.emitLABEL(labelCont)`

40. Cho khai báo sau được viết trên ngôn ngữ lập trình Ada:

```
type Shape is ( Circle , Triangle , Rectangle );  
type Colors is ( Red , Green , Blue );  
type Figure (Form: Shape) is record  
    Filled: Boolean;  
    Color: Colors;  
    case Form is  
        when Circle => Diameter: Float;  
        when Triangle =>  
            Leftside , Rightside: Integer;  
            Angle: Float;  
        when Rectangle => Side1 , Side2: Integer;  
    end case;  
end record;
```

Cho kích thước của các kiểu *Integer*, *Float*, *Boolean* và *Enumeration* lần lượt là 2, 4, 1, và 2, và kích thước cho phần mô tả kiểu (type description) là 0. Cho biết kích thước của một đối tượng kiểu **Figure**?

- (A) 21 (B) 15 (C) 13 (D) 23 (E) Khác

41. Cho biết kích thước của đối tượng dữ liệu **x** được khai báo như sau:
x: set of 1..16

- (A) 4 bytes (B) 2 bytes (C) 16 bytes (D) 4 bits (E) 1 byte

42. Cho $X = \{a, b\}$. Chọn biểu thức chính quy (regular expression) mô tả ngôn ngữ chứa bất kỳ chuỗi nào được tạo bởi các ký tự trên tập X nhưng KHÔNG chứa chuỗi có 2 ký tự **a** liên tiếp.

- (A) $a|(abb^*)^*$ (B) $b^*a?(bb^*a)^*b^*$ (C) $(b^*ab^*)^*$
(D) $b^*(abb^*)^*$ (E) $b^*ab^*ab^*$



43. Cho biết mã Jasmin của biểu thức viết bằng ngôn ngữ C như sau:

$a * b - 2$

với a và b là 2 biến nguyên có chỉ số lần lượt là 0 và 1.

	iload_0	iload_0	imul	iload_0	iload_0
	iload_1	iload_1	load_0	imul	iload_1
(A)	imul	(B)	iconst_2	(C)	iload_1
	iconst_2		imul	(D)	iload_1
	isub		isub	(E)	iconst_2
	isub		iconst_2		isub
		imul
					...

Đoạn mã sau được dùng trong các câu 44– 45:

```
int *p = new int;
void foo(int * r) {
    delete r;
}
foo(p); //1
*p = 2; //2
```

44. Hiện tượng gì xảy ra khi p được truyền cho r ở phát biểu //1 trong đoạn mã trên:

- (A) Con trỏ chưa khai báo (undeclared pointer)
 (B) Bí danh (alias) (C) Rác (garbage)
 (D) Khai báo trùng tên (redeclared) (E) Tham chiếu treo (dangling reference)

45. Hiện tượng gì xảy ra khi thực thi phát biểu //2 trong đoạn mã trên:

- (A) Tham chiếu treo (B) Bí danh (C) Con trỏ chưa khai báo
 (D) Đa hình (polymorphism) (E) Rác

46. Giả sử chương trình có một lỗi văn phạm nằm sau lệnh **print**. Khi em nhấn nút Run (để dịch và chạy chương trình) trên trình soạn thảo, kết quả được in ra bởi lệnh **print** trên trước khi lỗi văn phạm được báo. Hỏi chế độ dịch của trình soạn thảo là gì?

- (A) trình liên kết (link editor) (B) trình biên dịch (compiler)
 (C) trình biên dịch động(just-in-time compiler) (D) trình thông dịch(interpreter)
 (E) trình hợp ngữ (assembler)

47. Cho biết mã Jasmin của phát biểu gán sau:

$a[i] = a[i] + 2$

với i kiểu nguyên có chỉ số là 3 và a là kiểu dãy nguyên có chỉ số là 1.

	iastore		aload_1	aload_1	aload_1
	iadd		iload_3	iload_3	dup
	iaload		aload_1	aload_1	iload_3
(A)	aload_1	(B)	iconst_2	(C)	aload_1
	iload_3		iadd	(D)	iload_3
	dup		iastore	(E)	dup
	iconst_2		...		iaload
		iconst_2
			...		iadd
			...		iastore
		

II. Phần câu hỏi tự luận: (dành cho tất cả sinh viên) (2 điểm)

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

Lời giải. $(+ (- a (* b c d) e) f)$

Nếu viết đúng biểu thức dạng tiền tố Cambridge Polish: 1

Nếu viết đúng và có số () ít nhất: 1

III. Phần bài tập lớn: (dành cho tất cả sinh viên)

51. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện kiểm tra kiểu. Hàm `visitBlock` sẽ thực hiện chức năng xây dựng bảng danh hiệu (symbol table) và truyền bảng này khi visit các phát biểu trong khối. Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm. Nhắc lại AST của Block được khai báo như sau:
- ```
case class Block(val decl:List[Decl],val stmts:List[Stmt]) extends Stmt
```
52. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện sinh mã cho một phát biểu If. Nhắc lại AST của If được khai báo như sau:
- ```
case class If(val expr:Expr, val thenStmt:Stmt, val elseStmt:Option[Stmt]) extends Stmt
```
- Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm.

IV. Phần KSTN: (dành cho sinh viên KSTN)

53. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện kiểm tra lỗi FunctionNotReturn? Yêu cầu giải thích ý tưởng thực hiện thể hiện qua mã của hàm?
54. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện sinh mã cho một khối (block)? Yêu cầu giải thích ý tưởng thực hiện sinh mã cho các khai báo array trong một khối?

Question Distribution

Content	Level 1	Level 2	Level 3	Level 4
Introduction		1		
Lexical	1	4	5	
Syntax	2	4	7	
AST		2		1
Total	3	11	12	1

HẾT



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



KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH
BỘ MÔN KHOA HỌC MÁY TÍNH

Họ và tên:.....
MSSV:.....

Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình

Thời gian: 120 phút

Ngày thi: 23-12-2015

☐ Sinh viên được phép sử dụng tài liệu

☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1154

- Sinh viên phải ghi tên và mã số sinh viên trên đề thi (**trang đầu và trang cuối**), giấy làm bài trắc nghiệm (giấy đỏ) và giấy làm bài. Sinh viên phải tô phần mã đề thi và mã số sinh viên trên giấy làm bài trắc nghiệm. Khi nộp bài, sinh viên phải nộp cả **đề thi, giấy làm bài trắc nghiệm và giấy làm bài**.
- Phần trắc nghiệm sẽ được chấm TỰ ĐỘNG trên giấy làm bài trắc nghiệm. Do đó, phần trắc nghiệm nếu làm trên đề thi sẽ KHÔNG được chấm.
- Đối với các câu hỏi phần trắc nghiệm, sinh viên chỉ chọn MỘT phương án đúng nhất.
- Đối với phần câu hỏi tự luận (phần II), sinh viên làm ngay trên đề thi, ở phần dành riêng ngay dưới mỗi câu hỏi.
- Đối với câu hỏi bài tập lớn (phần III và IV), sinh viên làm trên giấy làm bài.
- Sinh viên lớp đại trà làm 3 phần (I, II và III). Sinh viên lớp tài năng làm cả 4 phần (I, II, III và IV). Chỉ có phần I và II được dùng để tính điểm cuối kỳ, các phần III và IV để tính điểm bài tập lớn 2 và 3.

I. Phần câu hỏi trắc nghiệm:(8 điểm))

1. Cho biết mã Jasmin của biểu thức viết bằng ngôn ngữ C như sau:

$a * b - 2$

với a và b là 2 biến nguyên có chỉ số lần lượt là 0 và 1.

iload_0	iload_0	iload_0	imul	iload_0
iload_1	iload_1	imul	load_0	iload_1
(A) imul	(B) iconst_2	(C) iload_1	(D) iload_1	(E) iconst_2
iconst_2	imul	isub	isub	isub
isub	isub	iconst_2	iconst_2	imul
...

Đoạn mã sau được dùng trong các câu 2– 3:

```
int *p = new int;
void foo(int * r) {
    delete r;
}
foo(p); //1
*p = 2; //2
```



2. Hiện tượng gì xảy ra khi p được truyền cho r ở phát biểu //1 trong đoạn mã trên:

- (A) Con trỏ chưa khai báo (undeclared pointer) (B) Bí danh (alias)
(C) Khai báo trùng tên (redeclared) (D) Rác (garbage)
(E) Tham chiếu treo (dangling reference)

3. Hiện tượng gì xảy ra khi thực thi phát biểu //2 trong đoạn mã trên:

- (A) Tham chiếu treo (B) Bí danh (C) Đa hình (polymorphism)
(D) Con trỏ chưa khai báo (E) Rác

4. Cho $X = \{a, b\}$. Chọn biểu thức chính quy (regular expression) mô tả ngôn ngữ chứa bất kỳ chuỗi nào được tạo bởi các ký tự trên tập X nhưng KHÔNG chứa chuỗi có 2 ký tự **a** liên tiếp.

- (A) $a|(abb^*)^*$ (B) $b^*(abb^*)^*$ (C) $(b^*ab^*)^*$ (D) $b^*a?(bb^*a)^*b^*$ (E) $b^*ab^*ab^*$

5. Cho biết kích thước của đối tượng dữ liệu x được khai báo như sau:
x: set of 1..16

- (A) 4 bytes (B) 2 bytes (C) 4 bits (D) 16 bytes (E) 1 byte

6. Chọn phát biểu ĐÚNG trong các phát biểu về dãy (array) sau:

- (A) Trên C, kích thước của dãy có thể tăng thêm sau khi dãy đã được tạo ra
(B) Truy xuất một phần tử bên ngoài một dãy C không được phép, sẽ lập tức gây ra lỗi
(C) Trên C và C++, kích thước của một dãy được lưu trữ để kiểm tra động
(D) Truy xuất một phần tử bên ngoài một dãy Java không được phép, sẽ lập tức gây ra lỗi
(E) Trên Java, kích thước của một dãy KHÔNG được lưu trữ để kiểm tra động

7. Trình định thời (scheduled subprograms) thường được dùng trong

- (A) Lập trình song song (Parallel Programming)
(B) Lập trình hướng đối tượng (Object-Oriented Programming)
(C) Lập trình hàm (Functional Programming)
(D) Lập trình thời gian thực (Real-time Programming)
(E) Lập trình hướng sự kiện (Event-driven Programming)

8. Liệt kê theo thứ tự chiều dài tăng dần ít nhất 5 chuỗi ngắn nhất (nếu có nhiều hơn 5 chuỗi) của ngôn ngữ được mô tả bởi biểu thức chính quy (regular expression) sau:
 $ab^*(a|b)?a$

- (A) aa, aba, aaa, abaa, aaba (B) ε, a, aa, aba, abaa (C) aa, aba, aaa, abaa, abba
(D) aa, aba, aaa, abaa, aaaa (E) a, aa, aba, aaa, aaba

Đoạn code sau, được viết trên ngôn ngữ dùng **qui tắc tầm vực tĩnh (static-scope rule)**, áp dụng cho các câu 9–12

```
procedure main(){  
  var a, b, c:integer; //1  
  procedure sub1(a: integer) { //2  
    procedure sub3();  
    procedure sub2() {  
      var a, c : real; //3  
      sub3();  
    }  
    procedure sub3() {
```



```
    a // use a
  }
  sub2();
}
sub1(3);
}
```

9. Môi trường tham khảo tĩnh (static referencing environment) của thủ tục **sub3** KHÔNG chứa
- (A) a ở //1 (B) sub1 (C) a ở //2 (D) sub2 (E) sub3
10. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong **sub3** ứng với khai báo
- (A) Báo lỗi a chưa khai báo (B) a trong sub1 (//2) (C) a trong sub2 (//3)
(D) a trong main (//1) (E) a trong sub3
11. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong sub3 **khi thực thi** ứng với kết hợp của **a** trong bản hoạt động của
- (A) main (B) sub1 (C) sub2 (D) Báo lỗi không tìm thấy a
(E) sub3
12. Nếu đoạn code trên được viết trên ngôn ngữ dùng qui tắc tầm vực động (dynamic-scope rule), và giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến a trong sub3 **khi thực thi** ứng với kết hợp của a trong bản hoạt động của
- (A) Báo lỗi không tìm thấy a (B) main (C) sub1 (D) sub3
(E) sub2
13. Đoạn code viết bằng Scala nào dưới đây có thể dùng như một ví dụ minh họa cho khái niệm Hàm Currying
- (A) `def add1(n:Int) = n + 1`
(B) `List(1,2,3).foldLeft(0)(_ + _)`
(C) `def add(n:Int) = (x:Int) => n + x; val add2 = add(2)`
(D) `List(1,2,3).filter(_ > 1)`
(E) `def add(n:Int)(x:Int) = n + x; val add2 = add (2) _`
14. Gọi append là hàm dùng để nối 2 danh sách thành 1 danh sách. Ví dụ `append([1,2,3],[4,5,6])` sẽ có kết quả là `[1,2,3,4,5,6]`. Hãy hiện thực hàm `append(a:List[Int],b:List[Int])` dùng hàm bậc cao (high-order function)?
- (A) `a.map(b)((x,y)=>x::y)` (B) `a.foldLeft(b)((x,y)=>x::y)`
(C) `a.foldRight(b)((x,y)=>x::y)` (D) `b.foldRight(a)((x,y)=>x::y)`
(E) `b.foldLeft(a)((x,y)=>x::y)`
15. Một biến trên ngôn ngữ Javascript có thể nhận những giá trị thuộc các kiểu khác nhau. Ví dụ:
- ```
var x = "def";
x = 1;
```
- Ràng buộc giữa một biến và kiểu của nó trên ngôn ngữ Javascript xảy ra trong thời gian
- (A) Hiện thực (implementation)      (B) Chạy (running)  
(C) Lập trình (programming)      (D) Khởi động (loading)  
(E) Dịch (compiling)





16. Kiểu *union* của ngôn ngữ C++ cho phép ghi vào ở thành phần này nhưng đọc lên ở thành phần khác như trong ví dụ sau:

```
union {
 int songuyen;
 char kytu[2];
} x;
x.songuyen = 12;
cout << x.kytu[0] << " " << x.kytu[1] << endl;
```

Đặc tính này của ngôn ngữ C++ gây ảnh hưởng XẤU NHẤT đối với

- (A) tính trực giao (orthogonality) (B) tính đơn giản (simplicity)  
(C) tính dễ viết (writability) (D) chi phí (cost)  
(E) tính tin cậy (reliability)

Phần trình bày sau dùng trong các câu hỏi 17– 18:

Cho mã giả của phát biểu *for var = expr1 to expr2 do body* như sau:

```
start: var = expr1
 etemp = expr2
loop: if (var >= etemp) goto out
 body
label1: var++
label2: goto loop
out:
```

17. Nếu trong thân *body* có thực thi lệnh *continue* thì điều khiển sẽ chuyển đến thực thi phát biểu có nhãn là:

- (A) out (B) start (C) loop (D) label2 (E) label1

18. Với hiện thực của phát biểu *for* như trên, phát biểu nào đặt trong thân phát biểu **for** dưới đây sẽ làm cho phát biểu này bị lặp mãi mãi?

```
n = 10; s = 0;
for i = s to n do _____
```

- (A)  $i = i - 1;$  (B)  $s = s - 1;$  (C)  $n = n + 1;$   
(D) câu A và C đúng  
(E) Không phát biểu nào trong các câu A, B và C làm phát biểu **for** lặp mãi mãi

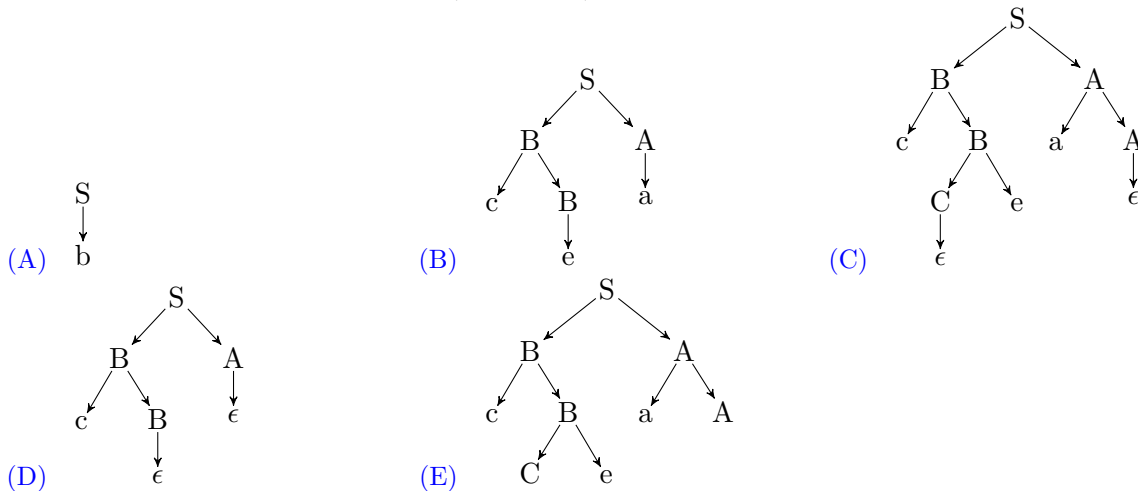
Cho văn phạm sau dùng cho các câu 19–20:

|   |   |     |  |            |
|---|---|-----|--|------------|
| S | → | b   |  | B A        |
| A | → | a A |  | $\epsilon$ |
| B | → | C e |  | c B        |
| C | → | d C |  | $\epsilon$ |

19. Chuỗi nào KHÔNG thuộc ngôn ngữ được mô tả bởi văn phạm trên?

- (A) e (B) b (C) eaaa (D) cccaaa (E) cccddddeaa

20. Chọn cây phân tích cú pháp (parse tree) cho chuỗi nhập: **cea**



21. Cơ chế gọi chương trình nào là cơ chế gọi chương trình con cơ bản của Lập trình hướng sự kiện (Event-driven Programming)

- (A) Biến cố - Xử lý biến cố (Exception)
- (B) Gọi trở về đơn giản (Simple Call - Return)
- (C) Gọi đệ qui (Recursive call)
- (D) Trình định thời (Scheduled Subprogram)
- (E) Trình cộng hành (Coroutine)

22. Khi cần khai thác đặc tính của kiểu union (các thành phần loại trừ lẫn nhau), làm thế nào để hiện thực trên những ngôn ngữ lập trình hướng đối tượng như Java (không có kiểu union)? Cụ thể làm thế nào để hiện thực biến z kiểu union sau trên những ngôn ngữ này?

```
union {
 string loikhen;
 string loiche;
} z;
```

- (A) Định nghĩa lớp cha A và hai lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A
- (B) Định nghĩa một lớp A có 2 thuộc tính loikhen và loiche; và z có kiểu A
- (C) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu A
- (D) Định nghĩa lớp trừu tượng (abstract class) A và 2 lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A
- (E) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu B

23. Giả sử chương trình có một lỗi văn phạm nằm sau lệnh **print**. Khi em nhấn nút Run (để dịch và chạy chương trình) trên trình soạn thảo, kết quả được in ra bởi lệnh **print** trước khi lỗi văn phạm được báo. Hỏi chế độ dịch của trình soạn thảo là gì?

- (A) trình liên kết (link editor)
- (B) trình biên dịch (compiler)
- (C) trình thông dịch (interpreter)
- (D) trình biên dịch động (just-in-time compiler)
- (E) trình hợp ngữ (assembler)

24. Cho lớp(class) A là lớp cha (superclass) của lớp B và B là lớp cha của hai lớp C và D. Trong lớp A, có khai báo phương thức thực thể (instance method) **foo** và phương thức này bị ghi đè(override) trên tất cả các lớp B, C và D. Cho biến x được khai báo như sau:  
**B x;**

Qui ước viết  $func_T$  là phương thức func được khai báo trong lớp T, cho biết tập của phương thức đích của cuộc gọi sau: **x.foo();**

- (A)  $\{foo_C, foo_D\}$  (B)  $\{foo_A, foo_B, foo_C, foo_D\}$  (C)  $\{foo_B\}$   
(D)  $\{foo_B, foo_C, foo_D\}$  (E)  $\{foo_A, foo_B\}$

25. Cơ chế gọi chương trình con nào mà điều khiển có thể chuyển vào một vị trí ở giữa chương trình con được gọi thay vì chuyển vào đầu chương trình con được gọi?

- (A) Trình định thời (Scheduled Subprogram)  
(B) Gọi trở về đơn giản (Simple Call - Return)  
(C) Gọi đệ qui (Recursive call)  
(D) Trình cộng hành (Coroutine)  
(E) Biến cố - Xử lý biến cố (Exception)

26. Cho biết mã Jasmin của phát biểu gán sau:

$a[i] = a[i] + 2$

với i kiểu nguyên có chỉ số là 3 và a là kiểu dãy nguyên có chỉ số là 1.

- |             |              |            |             |            |
|-------------|--------------|------------|-------------|------------|
|             |              |            |             |            |
|             |              |            |             | aload_1    |
| iastore     |              |            |             | dup        |
| iadd        | aload_1      |            |             | dup        |
| iaload      | iload_3      |            |             | iload_3    |
|             | iaload       |            |             | iastore    |
|             | iaload       |            |             | dup        |
| (A) aload_1 | (B) iconst_2 | (C) iaload | (D) aload_1 | (E) iaload |
| iload_3     | iadd         | iconst_2   | iload_3     | iconst_2   |
| dup         | iastore      | iadd       | iconst_2    | iadd       |
| iconst_2    | ...          | iastore    | iadd        | iastore    |
| ...         |              | ...        | ...         | ...        |

27. Cho giá trị ban đầu của biến **c** là 3, cho biết những giá trị **có thể có** của biến **a** sau khi thực hiện phép gán sau trên ngôn ngữ lập trình C?

$a = c * (c = 5);$

- (A) 15, 25, 9 (B) 15 (C) 25 (D) 15, 25 (E) 9

28. Viết lại biểu thức trung tố (infix) sau sang dạng hậu tố (postfix) Polish? Giả sử là tất cả các phép toán đều có 2 toán hạng và có độ ưu tiên và tính kết hợp như trong ngôn ngữ lập trình C.

$a + b * c - d * e + f$

- (A)  $a + b * c * d * e * - f +$  (B)  $a b + c * d - e * f +$  (C)  $a b c * + d - e f * +$   
(D)  $a b c * + d e * - f +$  (E)  $a b c * + d e * + f -$

Cho văn phạm của một biểu thức sau dùng cho các câu 29–31:

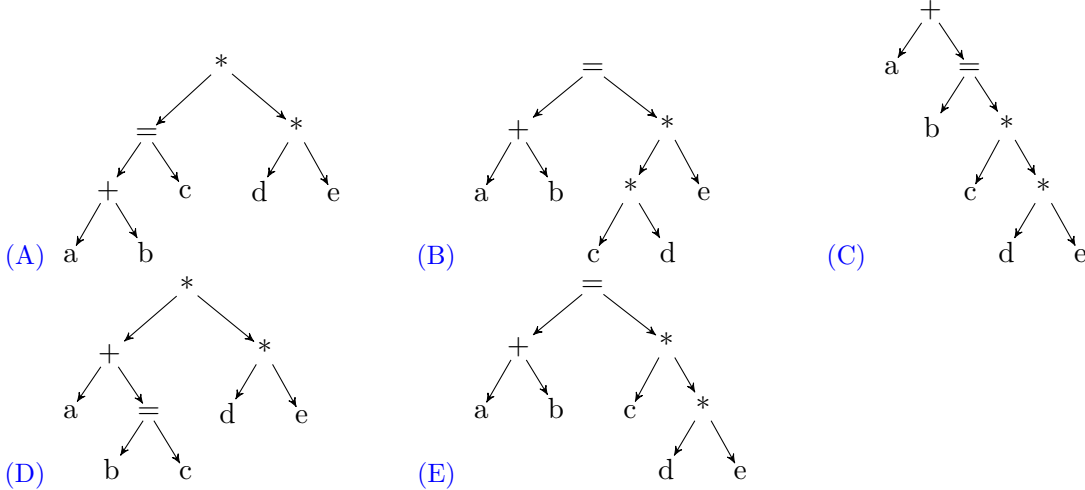
$exp \rightarrow term '=' exp \mid term$   
 $term \rightarrow term '+' fact \mid term '>' fact \mid fact$   
 $fact \rightarrow ope '**' ope \mid ope$   
 $ope \rightarrow '(' exp ')' \mid ID$

với ID là một danh hiệu.

29. Chọn biểu thức được viết đúng văn phạm, có số lượng dấu '(' và ')' ít nhất mà tương đương với biểu thức sau:  $(a + (b = ((c * d) * e))) > f$

- (A)  $(a + (b = (c * d) * e)) > f$  (B)  $(a + (b = ((c * d) * e))) > f$   
 (C)  $a + b = c * d * e > f$  (D)  $a + (b = c * d * e) > f$   
 (E)  $a + (b = (c * d) * e) > f$

30. Vẽ AST cho biểu thức sau được viết dựa vào văn phạm trên:  $a + b = c * (d * e)$



31. Cho cấu trúc AST được khai báo trên Scala như sau:

```
trait Exp
case class Bin(op:String,e1:Exp,e2:Exp) extends Exp
case class Id(i:String) extends Exp
```

Chọn AST cho biểu thức sau:  $a + b > c = d * e = f$

- (A) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Bin("=",Id("e"),Id("f"))))`  
 (B) `Bin("=",Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Id("e"))),Id("f"))`  
 (C) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("=",Bin("*",Id("d"),Id("e"))),Id("f"))`  
 (D) `Bin("=",Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("*",Id("d"),Id("e"))),Id("f"))`  
 (E) `Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("=",Bin("*",Id("d"),Id("e"))),Id("f"))`



32. Cho khai báo sau được viết trên ngôn ngữ lập trình Ada:

```

type Shape is (Circle , Triangle , Rectangle);
type Colors is (Red, Green, Blue);
type Figure (Form: Shape) is record
 Filled: Boolean;
 Color: Colors;
 case Form is
 when Circle => Diameter: Float;
 when Triangle =>
 Leftside, Rightside: Integer;
 Angle: Float;
 when Rectangle => Side1, Side2: Integer;
 end case;
end record;

```

Cho kích thước của các kiểu *Integer*, *Float*, *Boolean* và *Enumeration* lần lượt là 2, 4, 1, và 2, và kích thước cho phần mô tả kiểu (type description) là 0. Cho biết kích thước của một đối tượng kiểu **Figure**?

- (A) 21                      (B) 23                      (C) 13                      (D) 15                      (E) Khác

Đoạn code sau dùng cho các câu 33–35

```

int p;
int* foo(int x) {
 static int q;
 int *s = new int;
 switch (x) {
 case 1: return &p;
 case 2: return &q;
 case 3: return &x;
 case 4: return s;
 default: return foo(x-1);
 }
}

```

33. Phát biểu nào sẽ gây ra lỗi tham chiếu treo (dangling reference) khi thực thi  
 (A) return &p                      (B) return &x                      (C) return &q                      (D) return s  
 (E) Không phát biểu nào gây ra lỗi trên khi thực thi
34. Trong đoạn mã trên, đối tượng dữ liệu nào có thể trở thành rác (garbage)  
 (A) p                      (B) x                      (C) q                      (D) trở đến bởi s  
 (E) Không có đối tượng nào có thể trở thành rác
35. Khi hàm **foo** được gọi đệ quy (recursive) thì các bản hoạt động của **foo** dùng chung những đối tượng dữ liệu nào?  
 (A) q và s                      (B) p, q, x và s                      (C) p                      (D) x, q và s                      (E) p và q

Phần hướng dẫn này áp dụng cho các câu 36–40

Hãy viết thêm các đoạn mã cần thiết vào phương thức dưới đây để thực hiện sinh mã cho phát biểu *repeat*. Nhắc lại phát biểu *repeat* kết thúc quá trình lặp khi điều kiện của phát biểu trở thành **true**. AST của phát biểu **repeat** được định nghĩa như sau:

```

case class Repeat(val stmts:List[Stmt],val exp:Expr) extends Stmt

```



```
override def visitRepeat(ast: Repeat, o: Context) = {
 val ctxt = o.asInstanceOf[SubBody]
 ctxt.frame.enterLoop();
 val labelStart = ctxt.frame.getNewLabel()
 val labelBreak = ctxt.frame.getBreakLabel()
 val labelCont = ctxt.frame.getContinueLabel()
 val str1 = //1
 // sinh mã cho từng phát biểu trong thân của repeat
 val str2 = //2
 val str3 = //3
 // sinh mã cho biểu thức điều kiện
 val str4 = visit(ast.exp,...)
 val str5 = //4
 val str6 = //5
 frame().exitLoop();
 str1 + str2 + str3 + str4 + str5 + str6
}
```

36. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //1 trong đoạn mã trên

- |                                     |                                     |
|-------------------------------------|-------------------------------------|
| (A) ""                              | (B) ctxt.emit.emitLABEL(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitGOTO(labelStart)  |
| (E) ctxt.emit.emitLABEL(labelCont)  |                                     |

37. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //2 trong đoạn mã trên

- (A) visit(ast.stmts,o)  
(B) ast.stmts.map(x=>visit(x,o))  
(C) ast.stmts.filter(x=>visit(x,o))  
(D) ast.stmts.foldLeft("")(x,y)=>y + visit(x,o)  
(E) ast.stmts.foldLeft("")(x,y)=>x + visit(y,o)

38. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //3 trong đoạn mã trên

- |                                     |                                       |
|-------------------------------------|---------------------------------------|
| (A) ""                              | (B) ctxt.emit.emitIFFALSE(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitGOTO(labelStart)    |
| (E) ctxt.emit.emitLABEL(labelCont)  |                                       |

39. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //4 trong đoạn mã trên

- |                                     |                                       |
|-------------------------------------|---------------------------------------|
| (A) ctxt.emit.emitGOTO(labelBreak)  | (B) ctxt.emit.emitIFFALSE(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitGOTO(labelStart)    |
| (E) ctxt.emit.emitLABEL(labelCont)  |                                       |

40. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //5 trong đoạn mã trên

- |                                     |                                        |
|-------------------------------------|----------------------------------------|
| (A) ctxt.emit.emitGOTO(labelBreak)  | (B) ctxt.emit.emitIFFALSE(labelStart)) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitGOTO(labelStart))    |
| (E) ctxt.emit.emitLABEL(labelCont)) |                                        |

41. Để thực hiện tìm kiếm một chuỗi (String) trên nhiều loại danh sách khác nhau, hãy chọn giải pháp thích hợp nhất để điền vào chỗ trống trong định nghĩa hàm sau:

```
def lookup[T](x:String,lst:List[T],f:T=>String):Option[T] =
 lst match {
 case List() => None
 case h::t =>-----
 }
```

- (A) if (h == x) Some(h) else lookup(x,t,f)  
(B) if (f(h) == x) Some(h) else lookup(x,t,f)  
(C) if (h == f(x)) Some(h) else lookup(x,t,f)  
(D) if (h == f(x)) Some(f(h)) else lookup(x,t,f)  
(E) if (f(h) == x) Some(f(h)) else lookup(x,t,f)

Đoạn mã sau được dùng cho các câu 42–47.

Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:

```
int A[5] = {1,3,5,7,9}; // index of A starts from 0
int j = 0;
int n = 5;
int sumAndIncrease(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 = sumAndIncrease(A[j], j);
 printf("a = %i\n", s); //1
 printf(" %i %i %i %i %i\n", A[0], A[1], A[2], A[3], A[4]); //2
}
```

42. Nếu  $a$  và  $i$  được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến  $s$  được in ra ở phát biểu //1 là bao nhiêu?  
(A) 5 (B) 15 (C) 10 (D) 25 (E) Khác
43. Nếu  $a$  và  $i$  được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến  $A$  được in ra ở phát biểu //2 là bao nhiêu?  
(A) 1 3 5 7 9 (B) 6 3 5 7 9 (C) 2 4 6 8 10 (D) 5 3 5 7 9 (E) Khác
44. Nếu  $a$  và  $i$  được truyền bằng **tham khảo (passed by reference)**, giá trị của biến  $s$  được in ra ở phát biểu //1 là bao nhiêu?  
(A) 5 (B) 15 (C) 10 (D) 25 (E) Khác
45. Nếu  $a$  và  $i$  được truyền bằng **tham khảo (passed by reference)**, giá trị của biến  $A$  được in ra ở phát biểu //2 là bao nhiêu?  
(A) 1 3 5 7 9 (B) 6 3 5 7 9 (C) 2 4 6 8 10 (D) 5 3 5 7 9 (E) Khác



46. Nếu  $a$  và  $i$  được truyền bằng **tên (passed by name)**, giá trị của biến  $s$  được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5                      (B) 20                      (C) 15                      (D) 25                      (E) Khác

47. Nếu  $a$  và  $i$  được truyền bằng **tên (passed by name)**, giá trị của biến  $A$  được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9              (B) 6 3 5 7 9              (C) 2 4 6 8 10              (D) 5 3 5 7 9              (E) Khác

## II. Phần câu hỏi tự luận: (dành cho tất cả sinh viên) (2 điểm)

48. Trên những ngôn ngữ lập trình cho phép rút ngắn tính toán (short-circuit evaluation) biểu thức luận lý, biểu thức sau sẽ không gây ra lỗi khi thực thi. Biết hàm  $\text{sqrt}(a)$  trả về  $\sqrt{a}$  ( $(x < 0) \parallel (\text{sqrt}(x) > 4)$ )

Hãy dùng **if then else** để mô phỏng quá trình tính toán của biểu thức luận lý trên?

---

---

---

---

---

49. 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

---

---

50. Hãy nêu những giải pháp đã được sử dụng để tránh tham chiếu treo (dangling reference) trong trường hợp trả về hàm như sau:

```
void->int F() {
 int x = 1;
 int g() {
 return x + 1;
 }
 return g ;
}
void->int gg = F();
int z = gg();
```



This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

### III. Phần bài tập lớn: (dành cho tất cả sinh viên)

51. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện kiểm tra kiểu. Hàm `visitBlock` sẽ thực hiện chức năng xây dựng bảng danh hiệu (symbol table) và truyền bảng này khi visit các phát biểu trong khối. Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm. Nhắc lại AST của Block được khai báo như sau:

```
case class Block(val decls:List[Decl],val stmts:List[Stmt]) extends Stmt
```

52. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện sinh mã cho một phát biểu `If`. Nhắc lại AST của `If` được khai báo như sau:

```
case class If(val expr:Expr, val thenStmt:Stmt, val elseStmt:Option[Stmt]) extends Stmt
```

Yêu cầu giải thích rõ về `Context` cụ thể được sử dụng trong hàm và ý tưởng mã của hàm.

#### IV. Phần KSTN: (dành cho sinh viên KSTN)

53. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện kiểm tra lỗi `FunctionNotReTurn`? Yêu cầu giải thích ý tưởng thực hiện thể hiện qua mã của hàm?
54. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện sinh mã cho một khối (block)? Yêu cầu giải thích ý tưởng thực hiện sinh mã cho các khai báo array trong một khối?

-HẾT-

|                 |       |
|-----------------|-------|
| Họ và tên:..... | Điểm: |
| MSSV:.....      |       |



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



KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH  
BỘ MÔN KHOA HỌC MÁY TÍNH

## ĐÁP ÁN cho Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình

Thời gian: 120 phút

☐ Sinh viên được phép sử dụng tài liệu

☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1154

### I. Phần câu hỏi trắc nghiệm:(8 điểm))

1. Cho biết mã Jasmin của biểu thức viết bằng ngôn ngữ C như sau:

$a * b - 2$

với  $a$  và  $b$  là 2 biến nguyên có chỉ số lần lượt là 0 và 1.

|          |          |            |          |            |          |            |          |            |          |
|----------|----------|------------|----------|------------|----------|------------|----------|------------|----------|
|          | iload_0  |            | iload_0  |            | iload_0  |            | imul     |            | iload_0  |
|          | iload_1  |            | iload_1  |            | imul     |            | load_0   |            | iload_1  |
| <b>A</b> | imul     | <b>(B)</b> | iconst_2 | <b>(C)</b> | iload_1  | <b>(D)</b> | iload_1  | <b>(E)</b> | iconst_2 |
|          | iconst_2 |            | imul     |            | isub     |            | isub     |            | isub     |
|          | isub     |            | isub     |            | iconst_2 |            | iconst_2 |            | imul     |
|          | ...      |            | ...      |            | ...      |            | ...      |            | ...      |

Đoạn mã sau được dùng trong các câu 2– 3:

```
int *p = new int;
void foo(int * r) {
 delete r;
}
foo(p); //1
*p = 2; //2
```

2. Hiện tượng gì xảy ra khi  $p$  được truyền cho  $r$  ở phát biểu //1 trong đoạn mã trên:

- |                                                       |                                                 |
|-------------------------------------------------------|-------------------------------------------------|
| <b>(A)</b> Con trỏ chưa khai báo (undeclared pointer) | <b>(C)</b> Khai báo trùng tên (redeclared)      |
| <b>(B)</b> Bí danh (alias)                            | <b>(E)</b> Tham chiếu treo (dangling reference) |
| <b>(D)</b> Rác (garbage)                              |                                                 |

3. Hiện tượng gì xảy ra khi thực thi phát biểu //2 trong đoạn mã trên:

- |                                  |                    |                                   |
|----------------------------------|--------------------|-----------------------------------|
| <b>(A)</b> Tham chiếu treo       | <b>(B)</b> Bí danh | <b>(C)</b> Đa hình (polymorphism) |
| <b>(D)</b> Con trỏ chưa khai báo | <b>(E)</b> Rác     |                                   |

4. Cho  $X = \{a,b\}$ . Chọn biểu thức chính quy (regular expression) mô tả ngôn ngữ chứa bất kỳ chuỗi nào được tạo bởi các ký tự trên tập  $X$  nhưng KHÔNG chứa chuỗi có 2 ký tự  $a$  liên tiếp.

- |                          |                           |                          |                                |
|--------------------------|---------------------------|--------------------------|--------------------------------|
| <b>(A)</b> $a (abb^*)^*$ | <b>(B)</b> $b^*(abb^*)^*$ | <b>(C)</b> $(b^*ab^*)^*$ | <b>(D)</b> $b^*a?(bb^*a)^*b^*$ |
| <b>(E)</b> $b^*ab^*ab^*$ |                           |                          |                                |



5. Cho biết kích thước của đối tượng dữ liệu **x** được khai báo như sau:  
x: set of 1..16

- (A) 4 bytes      ☒ (B) 2 bytes      (C) 4 bits      (D) 16 bytes      (E) 1 byte

6. Chọn phát biểu ĐÚNG trong các phát biểu về dãy (array) sau:

- (A) Trên C, kích thước của dãy có thể tăng thêm sau khi dãy đã được tạo ra  
(B) Truy xuất một phần tử bên ngoài một dãy C không được phép, sẽ lập tức gây ra lỗi  
(C) Trên C và C++, kích thước của một dãy được lưu trữ để kiểm tra động  
☒ (D) Truy xuất một phần tử bên ngoài một dãy Java không được phép, sẽ lập tức gây ra lỗi  
(E) Trên Java, kích thước của một dãy KHÔNG được lưu trữ để kiểm tra động

7. Trình định thời (scheduled subprograms) thường được dùng trong

- (A) Lập trình song song (Parallel Programming)  
(B) Lập trình hướng đối tượng (Object-Oriented Programming)  
(C) Lập trình hàm (Functional Programming)  
☒ (D) Lập trình thời gian thực (Real-time Programming)  
(E) Lập trình hướng sự kiện (Event-driven Programming)

8. Liệt kê theo thứ tự chiều dài tăng dần ít nhất 5 chuỗi ngắn nhất (nếu có nhiều hơn 5 chuỗi) của ngôn ngữ được mô tả bởi biểu thức chính quy (regular expression) sau:  
 $ab^*(a|b)?a$

- (A) aa, aba, aaa, abaa, aaba      (B)  $\epsilon$ , a, aa, aba, abaa      ☒ (C) aa, aba, aaa, abaa, abba  
(D) aa, aba, aaa, abaa, aaaa      (E) a, aa, aba, aaa, aaba

Đoạn code sau, được viết trên ngôn ngữ dùng **qui tắc tầm vực tĩnh (static-scope rule)**, áp dụng cho các câu 9–12

```
procedure main(){
 var a, b, c:integer; //1
 procedure sub1(a: integer) { //2
 procedure sub3();
 procedure sub2() {
 var a, c : real; //3
 sub3();
 }
 procedure sub3() {
 a // use a
 }
 sub2();
 }
 sub1(3);
}
```

9. Môi trường tham khảo tĩnh (static referencing environment) của thủ tục **sub3** KHÔNG chứa

- ☒ (A) a ở //1      (B) sub1      (C) a ở //2      (D) sub2      (E) sub3



10. Giả sử chuỗi gọi là  $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$ , tham khảo đến **a** trong **sub3** ứng với khai báo
- (A) Báo lỗi a chưa khai báo      ☒ (B) a trong sub1 (//2)      (C) a trong sub2 (//3)  
(D) a trong main (//1)      ☒ (E) a trong sub3
11. Giả sử chuỗi gọi là  $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$ , tham khảo đến **a** trong sub3 **khi thực thi** ứng với kết hợp của **a** trong bản hoạt động của
- (A) main      ☒ (B) sub1      (C) sub2      (D) Báo lỗi không tìm thấy a  
(E) sub3
12. Nếu đoạn code trên được viết trên ngôn ngữ dùng qui tắc tầm vực động (dynamic-scope rule), và giả sử chuỗi gọi là  $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$ , tham khảo đến a trong sub3 **khi thực thi** ứng với kết hợp của a trong bản hoạt động của
- (A) Báo lỗi không tìm thấy a      (B) main      (C) sub1      (D) sub3  
☒ (E) sub2
13. Đoạn code viết bằng Scala nào dưới đây có thể dùng như một ví dụ minh họa cho khái niệm Hàm Currying
- (A) `def add1(n:Int) = n + 1`  
(B) `List(1,2,3).foldLeft(0)(_ + _)`  
(C) `def add(n:Int) = (x:Int) => n + x; val add2 = add(2)`  
(D) `List(1,2,3).filter(_ > 1)`  
☒ (E) `def add(n:Int)(x:Int) = n + x; val add2 = add (2) _`
14. Gọi append là hàm dùng để nối 2 danh sách thành 1 danh sách. Ví dụ `append([1,2,3],[4,5,6])` sẽ có kết quả là `[1,2,3,4,5,6]`. Hãy hiện thực hàm `append(a:List[Int],b:List[Int])` dùng hàm bậc cao (high-order function)?
- (A) `a.map(b)((x,y)=>x::y)`      (B) `a.foldLeft(b)((x,y)=>x::y)`  
☒ (C) `a.foldRight(b)((x,y)=>x::y)`      (D) `b.foldRight(a)((x,y)=>x::y)`  
(E) `b.foldLeft(a)((x,y)=>x::y)`
15. Một biến trên ngôn ngữ Javascript có thể nhận những giá trị thuộc các kiểu khác nhau. Ví dụ:
- ```
var x = "def";  
x = 1;
```
- Ràng buộc giữa một biến và kiểu của nó trên ngôn ngữ Javascript xảy ra trong thời gian
- (A) Hiện thực (implementation) ☒ (B) Chạy (running)
(C) Lập trình (programming) (D) Khởi động (loading)
(E) Dịch (compiling)



16. Kiểu *union* của ngôn ngữ C++ cho phép ghi vào ở thành phần này nhưng đọc lên ở thành phần khác như trong ví dụ sau:

```
union {
    int songuyen;
    char kytu[2];
} x;
x.songuyen = 12;
cout << x.kytu[0] << " " << x.kytu[1] << endl;
```

Đặc tính này của ngôn ngữ C++ gây ảnh hưởng XẤU NHẤT đối với

- (A) tính trực giao (orthogonality) (B) tính đơn giản (simplicity)
 (C) tính dễ viết (writability) (D) chi phí (cost)
☒ (E) tính tin cậy (reliability)

Phần trình bày sau dùng trong các câu hỏi 17– 18:

Cho mã giả của phát biểu *for var = expr1 to expr2 do body* như sau:

```
start:  var = expr1
        etemp = expr2
loop:   if (var >= etemp) goto out
        body
label1: var++
label2: goto loop
out:
```

17. Nếu trong thân *body* có thực thi lệnh *continue* thì điều khiển sẽ chuyển đến thực thi phát biểu có nhãn là:

- (A) out (B) start (C) loop (D) label2 ☒ (E) label1

18. Với hiện thực của phát biểu *for* như trên, phát biểu nào đặt trong thân phát biểu **for** dưới đây sẽ làm cho phát biểu này bị lặp mãi mãi?

```
n = 10; s = 0;
for i = s to n do _____
```

- ☒ (A) i = i - 1; (B) s = s - 1; (C) n = n + 1;
 (D) câu A và C đúng
 (E) Không phát biểu nào trong các câu A, B và C làm phát biểu **for** lặp mãi mãi

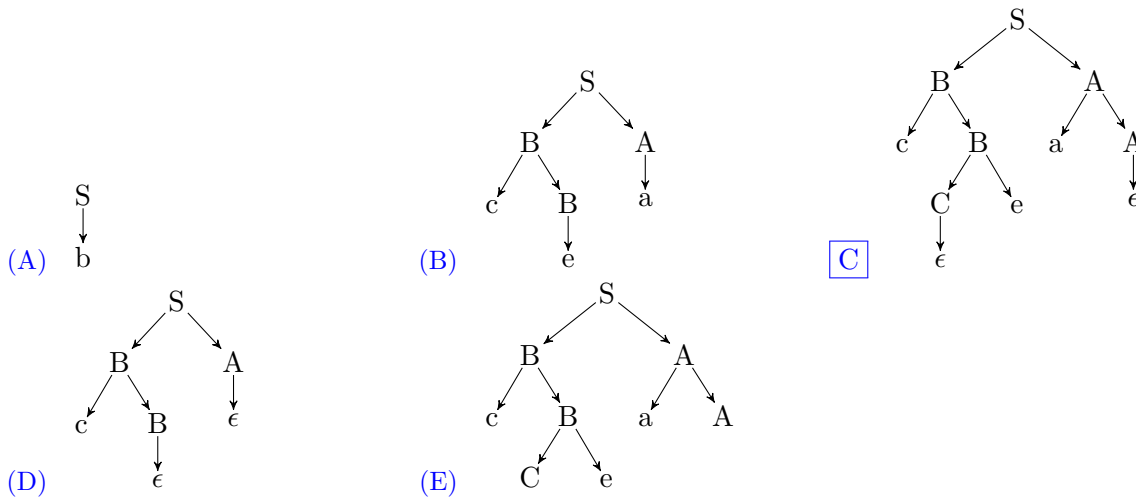
Cho văn phạm sau dùng cho các câu 19–20:

```
S → b | B A
A → a A | ε
B → C e | c B
C → d C | ε
```

19. Chuỗi nào KHÔNG thuộc ngôn ngữ được mô tả bởi văn phạm trên?

- (A) e (B) b (C) eaaa ☒ (D) cccaaa (E) cccddddeaaa

20. Chọn cây phân tích cú pháp (parse tree) cho chuỗi nhập: **cea**



21. Cơ chế gọi chương trình nào là cơ chế gọi chương trình con cơ bản của Lập trình hướng sự kiện (Event-driven Programming)

- ☒ (A) Biến cố - Xử lý biến cố (Exception)
- ☐ (B) Gọi trở về đơn giản (Simple Call - Return)
- ☐ (C) Gọi đệ qui (Recursive call)
- ☐ (D) Trình định thời (Scheduled Subprogram)
- ☐ (E) Trình cộng hành (Coroutine)

22. Khi cần khai thác đặc tính của kiểu union (các thành phần loại trừ lẫn nhau), làm thế nào để hiện thực trên những ngôn ngữ lập trình hướng đối tượng như Java (không có kiểu union)? Cụ thể làm thế nào để hiện thực biến z kiểu union sau trên những ngôn ngữ này?

```
union {
    string loikhen;
    string loiche;
} z;
```

- ☐ (A) Định nghĩa lớp cha A và hai lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A
- ☐ (B) Định nghĩa một lớp A có 2 thuộc tính loikhen và loiche; và z có kiểu A
- ☐ (C) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu A
- ☒ (D) Định nghĩa lớp trừu tượng (abstract class) A và 2 lớp con B (có thuộc tính loikhen) và C (có thuộc tính loiche), và z có kiểu A
- ☐ (E) Định nghĩa lớp A có thuộc tính loikhen và lớp B (con của A) có thuộc tính loiche, và z có kiểu B

23. Giả sử chương trình có một lỗi văn phạm nằm sau lệnh **print**. Khi em nhấn nút Run (để dịch và chạy chương trình) trên trình soạn thảo, kết quả được in ra bởi lệnh **print** trên trước khi lỗi văn phạm được báo. Hỏi chế độ dịch của trình soạn thảo là gì?

- ☐ (A) trình liên kết (link editor)
- ☐ (B) trình biên dịch (compiler)
- ☒ (C) trình thông dịch (interpreter)
- ☐ (D) trình biên dịch động (just-in-time compiler)
- ☐ (E) trình hợp ngữ (assembler)

24. Cho lớp(class) A là lớp cha (superclass) của lớp B và B là lớp cha của hai lớp C và D. Trong lớp A, có khai báo phương thức thực thể (instance method) **foo** và phương thức này bị ghi đè(override) trên tất cả các lớp B, C và D. Cho biến x được khai báo như sau: **B x;**

Qui ước viết $func_T$ là phương thức func được khai báo trong lớp T, cho biết tập của phương thức đích của cuộc gọi sau: **x.foo();**

- (A) $\{foo_C, foo_D\}$ (B) $\{foo_A, foo_B, foo_C, foo_D\}$ (C) $\{foo_B\}$
 (D) $\{foo_B, foo_C, foo_D\}$ (E) $\{foo_A, foo_B\}$

25. Cơ chế gọi chương trình con nào mà điều khiển có thể chuyển vào một vị trí ở giữa chương trình con được gọi thay vì chuyển vào đầu chương trình con được gọi?

- (A) Trình định thời (Scheduled Subprogram)
 (B) Gọi trở về đơn giản (Simple Call - Return)
 (C) Gọi đệ qui (Recursive call)
 (D) Trình cộng hành (Coroutine)
 (E) Biến cố - Xử lý biến cố (Exception)

26. Cho biết mã Jasmin của phát biểu gán sau:

$a[i] = a[i] + 2$

với i kiểu nguyên có chỉ số là 3 và a là kiểu dãy nguyên có chỉ số là 1.

- | | | | | | | | | | |
|-----|---|-----|--|-----|--|-----|--|-----|--|
| (A) | iastore
iadd
iaload
aload_1
iload_3
dup
iconst_2
... | (B) | aload_1
iload_3
iaload
iconst_2
iadd
iastore
... | (C) | aload_1
iload_3
aload_1
iload_3
iaload
iconst_2
iadd
iastore
... | (D) | aload_1
iload_3
iaload
iconst_2
iadd
iastore
... | (E) | aload_1
dup
iload_3
dup
iaload
iconst_2
iadd
iastore
... |
|-----|---|-----|--|-----|--|-----|--|-----|--|

27. Cho giá trị ban đầu của biến **c** là 3, cho biết những giá trị **có thể có** của biến **a** sau khi thực hiện phép gán sau trên ngôn ngữ lập trình C?

$a = c * (c = 5);$

- (A) 15, 25, 9 (B) 15 (C) 25 (D) 15, 25 (E) 9

28. Viết lại biểu thức trung tố (infix) sau sang dạng hậu tố (postfix) Polish? Giả sử là tất cả các phép toán đều có 2 toán hạng và có độ ưu tiên và tính kết hợp như trong ngôn ngữ lập trình C.

$a + b * c - d * e + f$

- (A) $a + b c * d e * - f +$ (B) $a b + c * d - e * f +$ (C) $a b c * + d - e f * +$
 (D) $a b c * + d e * - f +$ (E) $a b c * + d e * + f -$

Cho văn phạm của một biểu thức sau dùng cho các câu 29–31:

exp \rightarrow term '=' exp | term
 term \rightarrow term '+' fact | term '>' fact | fact
 fact \rightarrow ope '**' ope | ope
 ope \rightarrow '(' exp ')' | ID

với ID là một danh hiệu.

29. Chọn biểu thức được viết đúng văn phạm, có số lượng dấu '(' và ')' ít nhất mà tương đương với biểu thức sau: $(a + (b = ((c * d) * e))) > f$

(A) $(a + (b = (c * d) * e)) > f$

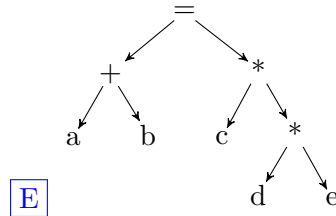
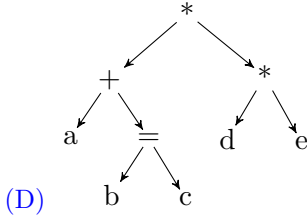
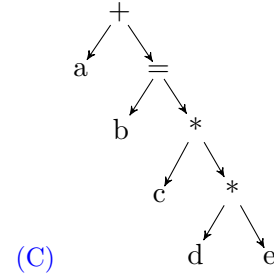
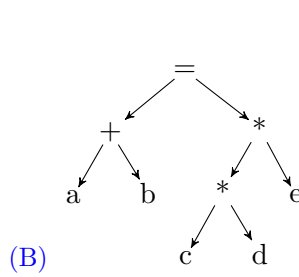
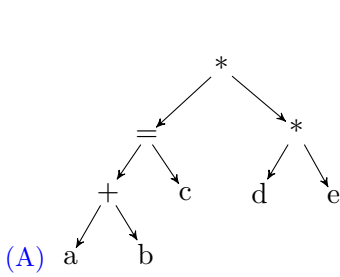
(B) $(a + (b = ((c * d) * e))) > f$

(C) $a + b = c * d * e > f$

(D) $a + (b = c * d * e) > f$

☒ (E) $a + (b = (c * d) * e) > f$

30. Vẽ AST cho biểu thức sau được viết dựa vào văn phạm trên: $a + b = c * (d * e)$



31. Cho cấu trúc AST được khai báo trên Scala như sau:

```
trait Exp
```

```
case class Bin(op:String,e1:Exp,e2:Exp) extends Exp
```

```
case class Id(i:String) extends Exp
```

Chọn AST cho biểu thức sau: $a + b > c = d * e = f$

(A) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Bin("=",Id("e"),Id("f"))))`

(B) `Bin("=",Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("*",Id("d"),Id("e"))),Id("f"))`

☒ (C) `Bin("=",Bin(">",Bin("+",Id("a"),Id("b")),Id("c")),Bin("=",Bin("*",Id("d"),Id("e"))),Id("f"))`

(D) `Bin("=",Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("*",Id("d"),Id("e"))),Id("f"))`

(E) `Bin("=",Bin("+",Id("a"),Bin(">",Id("b"),Id("c"))),Bin("=",Bin("*",Id("d"),Id("e"))),Id("f"))`



32. Cho khai báo sau được viết trên ngôn ngữ lập trình Ada:

```
type Shape is ( Circle , Triangle , Rectangle );
type Colors is ( Red, Green, Blue );
type Figure (Form: Shape) is record
    Filled: Boolean;
    Color: Colors;
    case Form is
        when Circle => Diameter: Float;
        when Triangle =>
            Leftside, Rightside: Integer;
            Angle: Float;
        when Rectangle => Side1, Side2: Integer;
    end case;
end record;
```

Cho kích thước của các kiểu *Integer*, *Float*, *Boolean* và *Enumeration* lần lượt là 2, 4, 1, và 2, và kích thước cho phần mô tả kiểu (type description) là 0. Cho biết kích thước của một đối tượng kiểu **Figure**?

- (A) 21 (B) 23 ☒ (C) 13 (D) 15 (E) Khác

Đoạn code sau dùng cho các câu 33–35

```
int p;
int* foo(int x) {
    static int q;
    int *s = new int;
    switch (x) {
        case 1: return &p;
        case 2: return &q;
        case 3: return &x;
        case 4: return s;
        default: return foo(x-1);
    }
}
```

33. Phát biểu nào sẽ gây ra lỗi tham chiếu treo (dangling reference) khi thực thi

- (A) return &p ☒ (B) return &x (C) return &q (D) return s
(E) Không phát biểu nào gây ra lỗi trên khi thực thi

34. Trong đoạn mã trên, đối tượng dữ liệu nào có thể trở thành rác (garbage)

- (A) p (B) x (C) q
☒ (D) trở đến bởi s (E) Không có đối tượng nào có thể trở thành rác

35. Khi hàm **foo** được gọi đệ quy (recursive) thì các bản hoạt động của **foo** dùng chung những đối tượng dữ liệu nào?

- (A) q và s (B) p, q, x và s (C) p (D) x, q và s ☒ (E) p và q

Phần hướng dẫn này áp dụng cho các câu 36–40



Hãy viết thêm các đoạn mã cần thiết vào phương thức dưới đây để thực hiện sinh mã cho phát biểu *repeat*. Nhắc lại phát biểu *repeat* kết thúc quá trình lặp khi điều kiện của phát biểu trở thành **true**. AST của phát biểu **repeat** được định nghĩa như sau:

```
case class Repeat(val stmts:List[Stmt],val exp:Expr) extends Stmt
```

```
override def visitRepeat(ast: Repeat, o:Context)= {  
    val ctxt = o.asInstanceOf[SubBody]  
    ctxt.frame.enterLoop();  
    val labelStart = ctxt.frame.getNewLabel()  
    val labelBreak = ctxt.frame.getBreakLabel()  
    val labelCont = ctxt.frame.getContinueLabel()  
    val str1 = //1  
    // sinh mã cho từng phát biểu trong thân của repeat  
    val str2 = //2  
    val str3 = //3  
    // sinh mã cho biểu thức điều kiện  
    val str4 = visit(ast.exp,...)  
    val str5 = //4  
    val str6 = //5  
    frame().exitLoop();  
    str1 + str2 + str3 + str4 + str5 + str6  
}
```

36. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //1 trong đoạn mã trên

- | | |
|-------------------------------------|-------------------------------------|
| (A) "" | (B) ctxt.emit.emitLABEL(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitGOTO(labelStart) |
| (E) ctxt.emit.emitLABEL(labelCont) | |

37. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //2 trong đoạn mã trên

- (A) visit(ast.stmts,o)
(B) ast.stmts.map(x=>visit(x,o))
(C) ast.stmts.filter(x=>visit(x,o))
(D) ast.stmts.foldLeft(")((x,y)=>y + visit(x,o))"
(E) ast.stmts.foldLeft(")((x,y)=>x + visit(y,o))"

38. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //3 trong đoạn mã trên

- | | |
|-------------------------------------|---------------------------------------|
| (A) "" | (B) ctxt.emit.emitIFFALSE(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitGOTO(labelStart) |
| (E) ctxt.emit.emitLABEL(labelCont) | |

39. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //4 trong đoạn mã trên

- | | |
|-------------------------------------|---------------------------------------|
| (A) ctxt.emit.emitGOTO(labelBreak) | (B) ctxt.emit.emitIFFALSE(labelStart) |
| (C) ctxt.emit.emitLABEL(labelBreak) | (D) ctxt.emit.emitGOTO(labelStart) |
| (E) ctxt.emit.emitLABEL(labelCont) | |

40. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //5 trong đoạn mã trên

- (A) `ctxt.emit.emitGOTO(labelBreak)` (B) `ctxt.emit.emitIFFALSE(labelStart)`
 (C) `ctxt.emit.emitLABEL(labelBreak)` (D) `ctxt.emit.emitGOTO(labelStart)`
 (E) `ctxt.emit.emitLABEL(labelCont)`

41. Để thực hiện tìm kiếm một chuỗi (String) trên nhiều loại danh sách khác nhau, hãy chọn giải pháp thích hợp nhất để điền vào chỗ trống trong định nghĩa hàm sau:

```
def lookup[T](x:String, lst:List[T], f:T=>String):Option[T] =
  lst match {
    case List() => None
    case h::t => _____
  }
```

- (A) `if (h == x) Some(h) else lookup(x,t,f)`
 (B) `if (f(h) == x) Some(h) else lookup(x,t,f)`
 (C) `if (h == f(x) Some(h) else lookup(x,t,f)`
 (D) `if (h == f(x)) Some(f(h)) else lookup(x,t,f)`
 (E) `if (f(h) == x) Some(f(h)) else lookup(x,t,f)`

Đoạn mã sau được dùng cho các câu 42–47.

Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:

```
int A[5] = {1,3,5,7,9}; // index of A starts from 0
int j = 0;
int n = 5;
int sumAndIncrease(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 = sumAndIncrease(A[j], j);
    printf("a = %i\n", s); //1
    printf(" %i %i %i %i %i\n", A[0], A[1], A[2], A[3], A[4]); //2
}
```

42. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 (B) 15 (C) 10 (D) 25 (E) Khác

43. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 6 3 5 7 9 (C) 2 4 6 8 10 (D) 5 3 5 7 9 (E) Khác



44. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 (B) 15 (C) 10 (D) 25 ☒ (E) Khác

45. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 6 3 5 7 9 ☒ (C) 2 4 6 8 10 (D) 5 3 5 7 9 (E) Khác

46. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 5 (B) 20 (C) 15 ☒ (D) 25 (E) Khác

47. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 1 3 5 7 9 (B) 6 3 5 7 9 ☒ (C) 2 4 6 8 10 (D) 5 3 5 7 9 (E) Khác

II. Phần câu hỏi tự luận: (dành cho tất cả sinh viên) (2 điểm)

48. Trên những ngôn ngữ lập trình cho phép rút ngắn tính toán (short-circuit evaluation) biểu thức luận lý, biểu thức sau sẽ không gây ra lỗi khi thực thi. Biết hàm $\text{sqrt}(a)$ trả về \sqrt{a} ($((x < 0) \parallel (\text{sqrt}(x) > 4))$)

Hãy dùng **if then else** để mô phỏng quá trình tính toán của biểu thức luận lý trên?

Lời giải. if $(x < 0)$ then ... else if $(\text{sqrt}(x) > 8)$ then ...

Giải pháp đề xuất đúng: 2

49. 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

Lời giải. $(+ (- a (* b c) (* d e) f)$

Nếu viết đúng biểu thức dạng tiền tố Cambridge Polish: 1

Nếu viết đúng và có số () ít nhất: 1

50. Hãy nêu những giải pháp đã được sử dụng để tránh tham chiếu treo (dangling reference) trong trường hợp trả về hàm như sau:

```
void->int F() {
    int x = 1;
    int g() {
        return x + 1;
    }
    return g ;
}
void->int gg = F();
int z = gg();
```

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

III. Phần bài tập lớn: (dành cho tất cả sinh viên)

51. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện kiểm tra kiểu. Hàm `visitBlock` sẽ thực hiện chức năng xây dựng bảng danh hiệu (symbol table) và truyền bảng này khi visit các phát biểu trong khối. Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm. Nhắc lại AST của Block được khai báo như sau:

```
case class Block(val decls:List[Decl],val stmts:List[Stmt]) extends Stmt
```

52. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện sinh mã cho một phát biểu `If`. Nhắc lại AST của `If` được khai báo như sau:

```
case class If(val expr:Expr, val thenStmt:Stmt, val elseStmt:Option[Stmt]) extends Stmt
```

Yêu cầu giải thích rõ về `Context` cụ thể được sử dụng trong hàm và ý tưởng mã của hàm.

IV. Phần KSTN: (dành cho sinh viên KSTN)

53. Viết hàm `visitIf(ast.If,o:Context)` để thực hiện kiểm tra lỗi `FunctionNotReTurn?` Yêu cầu giải thích ý tưởng thực hiện thể hiện qua mã của hàm?



54. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện sinh mã cho một khối (block)? Yêu cầu giải thích ý tưởng thực hiện sinh mã cho các khai báo array trong một khối?

Question Distribution

Content	Level 1	Level 2	Level 3	Level 4
Introduction		1		
Lexical	1	4	5	
Syntax	2	4	7	
AST		2		1
Total	3	11	12	1

HẾT

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



KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH
BỘ MÔN KHOA HỌC MÁY TÍNH

Họ và tên:.....
MSSV:.....

Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình

Thời gian: 120 phút

Ngày thi: 23-12-2015

☐ Sinh viên được phép sử dụng tài liệu

☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1155

- Sinh viên phải ghi tên và mã số sinh viên trên đề thi (**trang đầu và trang cuối**), giấy làm bài trắc nghiệm (giấy đỏ) và giấy làm bài. Sinh viên phải tô phần mã đề thi và mã số sinh viên trên giấy làm bài trắc nghiệm. Khi nộp bài, sinh viên phải nộp cả **đề thi, giấy làm bài trắc nghiệm và giấy làm bài**.
- Phần trắc nghiệm sẽ được chấm TỰ ĐỘNG trên giấy làm bài trắc nghiệm. Do đó, phần trắc nghiệm nếu làm trên đề thi sẽ KHÔNG được chấm.
- Đối với các câu hỏi phần trắc nghiệm, sinh viên chỉ chọn MỘT phương án đúng nhất.
- Đối với phần câu hỏi tự luận (phần II), sinh viên làm ngay trên đề thi, ở phần dành riêng ngay dưới mỗi câu hỏi.
- Đối với câu hỏi bài tập lớn (phần III và IV), sinh viên làm trên giấy làm bài.
- Sinh viên lớp đại trà làm 3 phần (I, II và III). Sinh viên lớp tài năng làm cả 4 phần (I, II, III và IV). Chỉ có phần I và II được dùng để tính điểm cuối kỳ, các phần III và IV để tính điểm bài tập lớn 2 và 3.

I. Phần câu hỏi trắc nghiệm:(8 điểm))

1. Cho biết mã Jasmin của phát biểu gán sau:

$a[i] = a[i] + 2$

với i kiểu nguyên có chỉ số là 3 và a là kiểu dãy nguyên có chỉ số là 1.

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Đoạn code sau, được viết trên ngôn ngữ dùng **qui tắc tầm vực tĩnh (static-scope rule)**, áp dụng cho các câu 2–5



```
procedure main(){
  var a, b, c:integer; //1
  procedure sub1(a: integer) { //2
    procedure sub3();
    procedure sub2() {
      var a, c : real;//3
      sub3();
    }
    procedure sub3() {
      a // use a
    }
    sub2();
  }
  sub1(3);
}
```

2. Môi trường tham khảo tĩnh (static referencing environment) của thủ tục **sub3** KHÔNG chứa
- (A) sub2 (B) a ở //1 (C) sub1 (D) a ở //2 (E) sub3
3. Giả sử chuỗi gọi là main → sub1 → sub2 → sub3, tham khảo đến **a** trong **sub3** ứng với khai báo
- (A) a trong sub1 (//2) (B) Báo lỗi a chưa khai báo (C) a trong sub3
(D) a trong sub2 (//3) (E) a trong main (//1)
4. Giả sử chuỗi gọi là main → sub1 → sub2 → sub3, tham khảo đến **a** trong sub3 **khi thực thi** ứng với kết hợp của **a** trong bản hoạt động của
- (A) sub1 (B) main (C) sub3 (D) sub2
(E) Báo lỗi không tìm thấy a
5. Nếu đoạn code trên được viết trên ngôn ngữ dùng qui tắc tầm vực động (dynamic-scope rule), và giả sử chuỗi gọi là main → sub1 → sub2 → sub3, tham khảo đến a trong sub3 **khi thực thi** ứng với kết hợp của a trong bản hoạt động của
- (A) main (B) Báo lỗi không tìm thấy a (C) sub2 (D) sub1
(E) sub3
6. Trình định thời (scheduled subprograms) thường được dùng trong
- (A) Lập trình hướng đối tượng (Object-Oriented Programming)
(B) Lập trình song song (Parallel Programming)
(C) Lập trình hướng sự kiện (Event-driven Programming)
(D) Lập trình hàm (Functional Programming)
(E) Lập trình thời gian thực (Real-time Programming)
7. Liệt kê theo thứ tự chiều dài tăng dần ít nhất 5 chuỗi ngắn nhất (nếu có nhiều hơn 5 chuỗi) của ngôn ngữ được mô tả bởi biểu thức chính quy (regular expression) sau:
ab*(a|b)? a
- (A) ε, a, aa, aba, abaa (B) aa, aba, aaa, abaa, aaba (C) a, aa, aba, aaa, aaba
(D) aa, aba, aaa, abaa, abba (E) aa, aba, aaa, abaa, aaaa

8. Giả sử chương trình có một lỗi văn phạm nằm sau lệnh **print**. Khi em nhấn nút Run (để dịch và chạy chương trình) trên trình soạn thảo, kết quả được in ra bởi lệnh **print** trên trước khi lỗi văn phạm được báo. Hỏi chế độ dịch của trình soạn thảo là gì?

- (A) trình biên dịch (compiler) (B) trình liên kết (link editor)
(C) trình hợp ngữ (assembler) (D) trình thông dịch(interpreter)
(E) trình biên dịch động(just-in-time compiler)

Đoạn mã sau được dùng trong các câu 9– 10:

```
int *p = new int;
void foo(int * r) {
    delete r;
}
foo(p); //1
*p = 2; //2
```

9. Hiện tượng gì xảy ra khi p được truyền cho r ở phát biểu //1 trong đoạn mã trên:

- (A) Bí danh (alias) (B) Con trỏ chưa khai báo (undeclared pointer) (C) Tham chiếu treo (dangling reference)
(D) Khai báo trùng tên (redeclared) (E) Rác (garbage)

10. Hiện tượng gì xảy ra khi thực thi phát biểu //2 trong đoạn mã trên:

- (A) Bí danh (B) Tham chiếu treo (C) Rác
(D) Đa hình (polymorphism) (E) Con trỏ chưa khai báo

Cho văn phạm của một biểu thức sau dùng cho các câu 11–13:

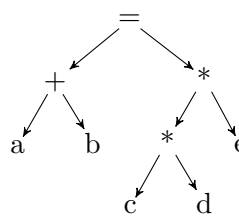
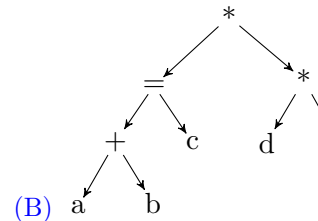
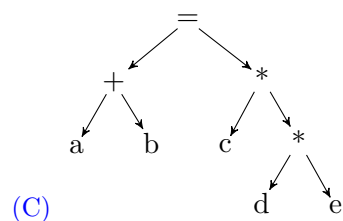
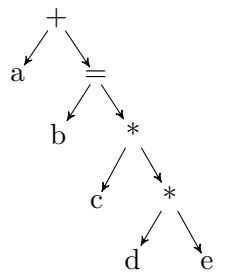
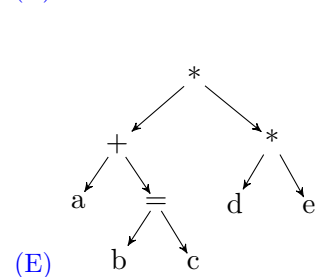
exp → term '=' exp | term
term → term '+' fact | term '>' fact | fact
fact → ope '*' ope | ope
ope → '(' exp ')' | ID

với ID là một danh hiệu.

11. Chọn biểu thức được viết đúng văn phạm, có số lượng dấu '(' và ')' ít nhất mà tương đương với biểu thức sau: $(a + (b = ((c * d) * e))) > f$

- (A) $(a + (b = ((c * d) * e))) > f$ (B) $(a + (b = (c * d) * e)) > f$
(C) $a + (b = (c * d) * e) > f$ (D) $a + b = c * d * e > f$
(E) $a + (b = c * d * e) > f$

12. Vẽ AST cho biểu thức sau được viết dựa vào văn phạm trên: $a + b = c * (d * e)$

- (A)  (B)  (C) 
(D)  (E) 



13. Cho cấu trúc AST được khai báo trên Scala như sau:

```
trait Exp
case class Bin(op:String,e1:Exp,e2:Exp) extends Exp
case class Id(i:String) extends Exp
```

Chọn AST cho biểu thức sau: $a + b > c = d * e = f$

- (A) $\text{Bin}("=", \text{Bin}("=", \text{Bin}(">", \text{Bin}("+", \text{Id}("a"), \text{Id}("b")), \text{Id}("c")), \text{Bin}("*", \text{Id}("d"), \text{Id}("e")), \text{Id}("f")))$
 (B) $\text{Bin}("=", \text{Bin}(">", \text{Bin}("+", \text{Id}("a"), \text{Id}("b")), \text{Id}("c")), \text{Bin}("*", \text{Id}("d"), \text{Bin}("=", \text{Id}("e"), \text{Id}("f"))))$
 (C) $\text{Bin}("=", \text{Bin}("+", \text{Id}("a"), \text{Bin}(">", \text{Id}("b"), \text{Id}("c"))), \text{Bin}("=", \text{Bin}("*", \text{Id}("d"), \text{Id}("e")), \text{Id}("f")))$
 (D) $\text{Bin}("=", \text{Bin}(">", \text{Bin}("+", \text{Id}("a"), \text{Id}("b")), \text{Id}("c")), \text{Bin}("=", \text{Bin}("*", \text{Id}("d"), \text{Id}("e")), \text{Id}("f")))$
 (E) $\text{Bin}("=", \text{Bin}("=", \text{Bin}("+", \text{Id}("a"), \text{Bin}(">", \text{Id}("b"), \text{Id}("c"))), \text{Bin}("*", \text{Id}("d"), \text{Id}("e")), \text{Id}("f")))$

14. Cho giá trị ban đầu của biến c là 3, cho biết những giá trị **có thể có** của biến a sau khi thực hiện phép gán sau trên ngôn ngữ lập trình C?

$a = c * (c = 5);$

- (A) 15 (B) 15, 25, 9 (C) 9 (D) 25 (E) 15, 25

15. Viết lại biểu thức trung tố (infix) sau sang dạng hậu tố (postfix) Polish? Giả sử là tất cả các phép toán đều có 2 toán hạng và có độ ưu tiên và tính kết hợp như trong ngôn ngữ lập trình C.

$a + b * c - d * e + f$

- (A) $a b + c * d - e * f +$ (B) $a + b c * d e * - f +$ (C) $a b c * + d e * + f -$
 (D) $a b c * + d - e f * +$ (E) $a b c * + d e * - f +$

Cho văn phạm sau dùng cho các câu 16–17:

$S \rightarrow b \mid B A$
 $A \rightarrow a A \mid \epsilon$
 $B \rightarrow C e \mid c B$
 $C \rightarrow d C \mid \epsilon$

16. Chuỗi nào KHÔNG thuộc ngôn ngữ được mô tả bởi văn phạm trên?

- (A) b (B) e (C) cccdddeaaa (D) eaaa (E) cccaaa

17. Chọn cây phân tích cú pháp (parse tree) cho chuỗi nhập: **cea**

- (A)
 (B)
 (C)
 (D)
 (E)

18. Cho biết mã Jasmin của biểu thức viết bằng ngôn ngữ C như sau:

$a * b - 2$

với a và b là 2 biến nguyên có chỉ số lần lượt là 0 và 1.

	iload_0	iload_0	iload_0	iload_0	imul
	iload_1	iload_1	iload_1	imul	load_0
(A) iconst_2	(B) imul	(C) iconst_2	(D) iload_1	(E) iload_1	
imul	iconst_2	isub	isub	isub	
isub	isub	imul	iconst_2	iconst_2	
...

19. Cho biết kích thước của đối tượng dữ liệu x được khai báo như sau:

x : set of 1..16

- (A) 2 bytes (B) 4 bytes (C) 1 byte (D) 4 bits (E) 16 bytes

20. Để thực hiện tìm kiếm một chuỗi (String) trên nhiều loại danh sách khác nhau, hãy chọn giải pháp thích hợp nhất để điền vào chỗ trống trong định nghĩa hàm sau:

```
def lookup[T](x:String, lst:List[T], f:T=>String):Option[T] =
  lst match {
    case List() => None
    case h::t => _____
  }
```

- (A) if (f(h) == x) Some(h) else lookup(x,t,f) (B) if (h == x) Some(h) else lookup(x,t,f)
 (C) if (f(h) == x) Some(f(h)) else lookup(x,t,f)
 (D) if (h == f(x)) Some(h) else lookup(x,t,f)
 (E) if (h == f(x)) Some(f(h)) else lookup(x,t,f)

21. Chọn phát biểu ĐÚNG trong các phát biểu về dãy (array) sau:

- (A) Truy xuất một phần tử bên ngoài một dãy C không được phép, sẽ lập tức gây ra lỗi
 (B) Trên C, kích thước của dãy có thể tăng thêm sau khi dãy đã được tạo ra
 (C) Trên Java, kích thước của một dãy KHÔNG được lưu trữ để kiểm tra động
 (D) Trên C và C++, kích thước của một dãy được lưu trữ để kiểm tra động
 (E) Truy xuất một phần tử bên ngoài một dãy Java không được phép, sẽ lập tức gây ra lỗi

22. Cơ chế gọi chương trình con nào mà điều khiển có thể chuyển vào một vị trí ở giữa chương trình con được gọi thay vì chuyển vào đầu chương trình con được gọi?

- (A) Gọi trở về đơn giản (Simple Call - Return)
 (B) Trình định thời (Scheduled Subprogram)
 (C) Biến cố - Xử lý biến cố (Exception)
 (D) Gọi đệ qui (Recursive call)
 (E) Trình cộng hành (Coroutine)

Phần hướng dẫn này áp dụng cho các câu 23–27

Hãy viết thêm các đoạn mã cần thiết vào phương thức dưới đây để thực hiện sinh mã cho phát biểu *repeat*. Nhắc lại phát biểu *repeat* kết thúc quá trình lặp khi điều kiện của phát biểu trở thành **true**. AST của phát biểu **repeat** được định nghĩa như sau:

```
case class Repeat(val stmts:List[Stmt],val exp:Expr) extends Stmt
```

```
override def visitRepeat(ast: Repeat, o:Context)= {
  val ctxt = o.asInstanceOf[SubBody]
```

```
ctxt.frame.enterLoop();
val labelStart = ctxt.frame.getNewLabel()
val labelBreak = ctxt.frame.getBreakLabel()
val labelCont = ctxt.frame.getContinueLabel()
val str1 = //1
// sinh mã cho từng phát biểu trong thân của repeat
val str2 = //2
val str3 = //3
// sinh mã cho biểu thức điều kiện
val str4 = visit(ast.exp,...)
val str5 = //4
val str6 = //5
frame().exitLoop();
str1 + str2 + str3 + str4 + str5 + str6
}
```

23. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //1 trong đoạn mã trên

- (A) `ctxt.emit.emitLABEL(labelStart)` (B) `""`
(C) `ctxt.emit.emitLABEL(labelCont)` (D) `ctxt.emit.emitLABEL(labelBreak)`
(E) `ctxt.emit.emitGOTO(labelStart)`

24. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //2 trong đoạn mã trên

- (A) `ast.stmts.map(x=>visit(x,o))`
(B) `visit(ast.stmts,o)`
(C) `ast.stmts.foldLeft(")((x,y)=>x + visit(y,o))`
(D) `ast.stmts.filter(x=>visit(x,o))`
(E) `ast.stmts.foldLeft(")((x,y)=>y + visit(x,o))`

25. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //3 trong đoạn mã trên

- (A) `ctxt.emit.emitIFFALSE(labelStart)` (B) `""`
(C) `ctxt.emit.emitLABEL(labelCont)` (D) `ctxt.emit.emitLABEL(labelBreak)`
(E) `ctxt.emit.emitGOTO(labelStart)`

26. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //4 trong đoạn mã trên

- (A) `ctxt.emit.emitIFFALSE(labelStart)` (B) `ctxt.emit.emitGOTO(labelBreak)`
(C) `ctxt.emit.emitLABEL(labelCont)` (D) `ctxt.emit.emitLABEL(labelBreak)`
(E) `ctxt.emit.emitGOTO(labelStart)`

27. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //5 trong đoạn mã trên

- (A) `ctxt.emit.emitIFFALSE(labelStart))` (B) `ctxt.emit.emitGOTO(labelBreak))`
(C) `ctxt.emit.emitLABEL(labelCont))` (D) `ctxt.emit.emitLABEL(labelBreak))`
(E) `ctxt.emit.emitGOTO(labelStart))`

28. Cho $X = \{a,b\}$. Chọn biểu thức chính quy (regular expression) mô tả ngôn ngữ chứa bất kỳ chuỗi nào được tạo bởi các ký tự trên tập X nhưng KHÔNG chứa chuỗi có 2 ký tự a liên tiếp.

- (A) `b*a?(bb*a)*b*` (B) `a|(abb*)*` (C) `b*(abb*)*` (D) `(b*ab*)*` (E) `b*ab*ab*`



Đoạn code sau dùng cho các câu 29–31

```
int p;  
int* foo(int x) {  
    static int q;  
    int *s = new int;  
    switch (x) {  
        case 1: return &p;  
        case 2: return &q;  
        case 3: return &x;  
        case 4: return s;  
        default: return foo(x-1);  
    }  
}
```

29. Phát biểu nào sẽ gây ra lỗi tham chiếu treo (dangling reference) khi thực thi
(A) return s (B) return &p (C) return &x (D) return &q
(E) Không phát biểu nào gây ra lỗi trên khi thực thi
30. Trong đoạn mã trên, đối tượng dữ liệu nào có thể trở thành rác (garbage)
(A) trỏ đến bởi s (B) p (C) x (D) q
(E) Không có đối tượng nào có thể trở thành rác
31. Khi hàm **foo** được gọi đệ quy (recursive) thì các bản hoạt động của **foo** dùng chung những đối tượng dữ liệu nào?
(A) p, q, x và s (B) q và s (C) p và q (D) p (E) x, q và s
32. Cho khai báo sau được viết trên ngôn ngữ lập trình Ada:

```
type Shape is ( Circle , Triangle , Rectangle );  
type Colors is ( Red , Green , Blue );  
type Figure (Form: Shape) is record  
    Filled: Boolean;  
    Color: Colors;  
    case Form is  
        when Circle => Diameter: Float;  
        when Triangle =>  
            Leftside , Rightside: Integer;  
            Angle: Float;  
        when Rectangle => Side1 , Side2: Integer;  
    end case;  
end record;
```

Cho kích thước của các kiểu *Integer*, *Float*, *Boolean* và *Enumeration* lần lượt là 2, 4, 1, và 2, và kích thước cho phần mô tả kiểu (type description) là 0. Cho biết kích thước của một đối tượng kiểu **Figure**?

- (A) 15 (B) 21 (C) 23 (D) 13 (E) Khác

Đoạn mã sau được dùng cho các câu 33–38.

Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:



```
int A[5] = {1,3,5,7,9}; // index of A starts from 0
int j = 0;
int n = 5;
int sumAndIncrease(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 = sumAndIncrease(A[j], j);
    printf("a = %i\n", s); //1
    printf(" %i %i %i %i %i\n", A[0], A[1], A[2], A[3], A[4]); //2
}
```

33. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?
(A) 25 (B) 5 (C) 15 (D) 10 (E) Khác
34. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?
(A) 5 3 5 7 9 (B) 1 3 5 7 9 (C) 6 3 5 7 9 (D) 2 4 6 8 10 (E) Khác
35. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?
(A) 25 (B) 5 (C) 15 (D) 10 (E) Khác
36. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?
(A) 5 3 5 7 9 (B) 1 3 5 7 9 (C) 6 3 5 7 9 (D) 2 4 6 8 10 (E) Khác
37. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?
(A) 25 (B) 5 (C) 20 (D) 15 (E) Khác
38. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?
(A) 5 3 5 7 9 (B) 1 3 5 7 9 (C) 6 3 5 7 9 (D) 2 4 6 8 10 (E) Khác
39. Cho lớp(class) A là lớp cha (superclass) của lớp B và B là lớp cha của hai lớp C và D . Trong lớp A , có khai báo phương thức thực thể (instance method) **foo** và phương thức này bị ghi đè(override) trên tất cả các lớp B , C và D . Cho biến x được khai báo như sau:
B x;
- Qui ước viết $func_T$ là phương thức func được khai báo trong lớp T , cho biết tập của phương thức đích của cuộc gọi sau: **x.foo();**
- (A) $\{foo_A, foo_B, foo_C, foo_D\}$ (B) $\{foo_C, foo_D\}$ (C) $\{foo_A, foo_B\}$
(D) $\{foo_B\}$ (E) $\{foo_B, foo_C, foo_D\}$

40. Cơ chế gọi chương trình nào là cơ chế gọi chương trình con cơ bản của Lập trình hướng sự kiện (Event-driven Programming)

- (A) Gọi trở về đơn giản (Simple Call - Return)
- (B) Biến cố - Xử lý biến cố (Exception)
- (C) Trình cộng hành (Coroutine)
- (D) Gọi đệ qui (Recursive call)
- (E) Trình định thời (Scheduled Subprogram)

41. Một biến trên ngôn ngữ Javascript có thể nhận những giá trị thuộc các kiểu khác nhau. Ví dụ:

```
var x = "def";  
x = 1;
```

Ràng buộc giữa một biến và kiểu của nó trên ngôn ngữ Javascript xảy ra trong thời gian

- (A) Chạy (running)
- (B) Hiện thực (implementation)
- (C) Dịch (compiling)
- (D) Lập trình (programming)
- (E) Khởi động (loading)

42. Gọi append là hàm dùng để nối 2 danh sách thành 1 danh sách. Ví dụ `append([1,2,3],[4,5,6])` sẽ có kết quả là `[1,2,3,4,5,6]`. Hãy hiện thực hàm `append(a:List[Int],b:List[Int])` dùng hàm bậc cao (high-order function)?

- (A) `a.foldLeft(b)((x,y)=>x::y)`
- (B) `a.map(b)((x,y)=>x::y)`
- (C) `b.foldLeft(a)((x,y)=>x::y)`
- (D) `a.foldRight(b)((x,y)=>x::y)`
- (E) `b.foldRight(a)((x,y)=>x::y)`

Phần trình bày sau dùng trong các câu hỏi 43– 44:

Cho mã giả của phát biểu *for var = expr1 to expr2 do body* như sau:

```
start:  var = expr1  
        etemp = expr2  
loop:   if (var >= etemp) goto out  
        body  
label1: var++  
label2: goto loop  
out:
```

43. Nếu trong thân *body* có thực thi lệnh *continue* thì điều khiển sẽ chuyển đến thực thi phát biểu có nhãn là:

- (A) start
- (B) out
- (C) label1
- (D) loop
- (E) label2

44. Với hiện thực của phát biểu *for* như trên, phát biểu nào đặt trong thân phát biểu **for** dưới đây sẽ làm cho phát biểu này bị lặp mãi mãi?

```
n = 10; s = 0;  
for i = s to n do -----
```

- (A) `i = i - 1;`
- (B) `s = s - 1;`
- (C) `n = n + 1;`
- (D) câu A và C đúng
- (E) Không phát biểu nào trong các câu A, B và C làm phát biểu **for** lặp mãi mãi

45. Khi cần khai thác đặc tính của kiểu union (các thành phần loại trừ lẫn nhau), làm thế nào để hiện thực trên những ngôn ngữ lập trình hướng đối tượng như Java (không có kiểu union)? Cụ thể làm thế nào để hiện thực biến `z` kiểu union sau trên những ngôn ngữ này?

```
union {  
    string loikhen;  
    string loiche;  
} z;
```

- (A) Định nghĩa một lớp `A` có 2 thuộc tính `loikhen` và `loiche`; và `z` có kiểu `A`
- (B) Định nghĩa lớp cha `A` và hai lớp con `B` (có thuộc tính `loikhen`) và `C` (có thuộc tính `loiche`), và `z` có kiểu `A`
- (C) Định nghĩa lớp `A` có thuộc tính `loikhen` và lớp `B` (con của `A`) có thuộc tính `loiche`, và `z` có kiểu `B`
- (D) Định nghĩa lớp `A` có thuộc tính `loikhen` và lớp `B` (con của `A`) có thuộc tính `loiche`, và `z` có kiểu `A`
- (E) Định nghĩa lớp trừu tượng (abstract class) `A` và 2 lớp con `B` (có thuộc tính `loikhen`) và `C` (có thuộc tính `loiche`), và `z` có kiểu `A`

46. Đoạn code viết bằng Scala nào dưới đây có thể dùng như một ví dụ minh họa cho khái niệm Hàm Currying

- (A) `List(1,2,3).foldLeft(0)(_ + _)`
- (B) `def add1(n:Int) = n + 1`
- (C) `def add(n:Int)(x:Int) = n + x; val add2 = add (2) _`
- (D) `def add(n:Int) = (x:Int) => n + x; val add2 = add(2)`
- (E) `List(1,2,3).filter(_ > 1)`

47. Kiểu *union* của ngôn ngữ C++ cho phép ghi vào ở thành phần này nhưng đọc lên ở thành phần khác như trong ví dụ sau:

```
union {  
    int songuyen;  
    char kytu[2];  
} x;  
x.songuyen = 12;  
cout << x.kytu[0] << " " << x.kytu[1] << endl;
```

Đặc tính này của ngôn ngữ C++ gây ảnh hưởng XẤU NHẤT đối với

- (A) tính đơn giản (simplicity)
- (B) tính trực giao (orthogonality)
- (C) tính tin cậy (reliability)
- (D) tính dễ viết (writability)
- (E) chi phí (cost)

II. Phần câu hỏi tự luận: (dành cho tất cả sinh viên) (2 điểm)

48. 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:



51. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện kiểm tra kiểu. Hàm `visitBlock` sẽ thực hiện chức năng xây dựng bảng danh hiệu (symbol table) và truyền bảng này khi visit các phát biểu trong khối. Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm. Nhắc lại AST của Block được khai báo như sau:
`case class Block(val decl:List[Decl],val stmts:List[Stmt]) extends Stmt`

52. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện sinh mã cho một phát biểu If. Nhắc lại AST của If được khai báo như sau:
`case class If(val expr:Expr, val thenStmt:Stmt, val elseStmt:Option[Stmt]) extends Stmt`
Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm.

IV. Phần KSTN: (dành cho sinh viên KSTN)

53. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện kiểm tra lỗi `FunctionNotReTurn`? Yêu cầu giải thích ý tưởng thực hiện thể hiện qua mã của hàm?
54. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện sinh mã cho một khối (block)? Yêu cầu giải thích ý tưởng thực hiện sinh mã cho các khai báo array trong một khối?

HẾT

Họ và tên:.....	Điểm:
MSSV:.....	



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

KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH
BỘ MÔN KHOA HỌC MÁY TÍNH

ĐÁP ÁN cho Đề thi cuối kỳ

Môn thi: Nguyên Lý Ngôn Ngữ Lập Trình

Thời gian: 120 phút

□ Sinh viên được phép sử dụng tài liệu

☒ Sinh viên không được sử dụng tài liệu

Mã đề: 1155

I. Phần câu hỏi trắc nghiệm:(8 điểm))

1. Cho biết mã Jasmin của phát biểu gán sau:

$$a[i] = a[i] + 2$$

với i kiểu nguyên có chỉ số là 3 và a là kiểu dãy nguyên có chỉ số là 1.

(A)	aload_1 iload_3 iaload iconst_2 iadd iastore ...	(B)	iastore iadd iaload aload_1 iload_3 dup iconst_2 ...	(C)	aload_1 dup iload_3 dup iaload iconst_2 iadd iastore ...	(D)	aload_1 iload_3 aload_1 iload_3 iaload iconst_2 iadd iastore ...	(E)	aload_1 iload_3 iastore aload_1 iload_3 iconst_2 iadd ...
-----	--	-----	---	-----	--	-----	--	-----	--

Đoạn code sau, được viết trên ngôn ngữ dùng **qui tắc tầm vực tĩnh** (static-scope rule), áp dụng cho các câu 2-5

```

procedure main(){
    var a, b, c:integer; //1
    procedure sub1(a: integer) { //2
        procedure sub3();
        procedure sub2() {
            var a, c : real;//3
            sub3();
        }
        procedure sub3() {
            a // use a
        }
        sub2();
    }
    sub1(3);
}

```

2. Môi trường tham khảo tĩnh (static referencing environment) của thủ tục **sub3** KHÔNG chứa

$$(A) \text{ sub2} \quad \boxed{B} \text{ a } \mathfrak{O} // 1 \quad (C) \text{ sub1} \quad (D) \text{ a } \mathfrak{O} // 2 \quad (E) \text{ sub3}$$



3. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong **sub3** ứng với khai báo

- ☐ (A) a trong sub1 (//2) ☐ (B) Báo lỗi a chưa khai báo ☐ (C) a trong sub3
☐ (D) a trong sub2 (//3) ☐ (E) a trong main (//1)

4. Giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến **a** trong sub3 **khi thực thi** ứng với kết hợp của **a** trong bản hoạt động của

- ☐ (A) sub1 ☐ (B) main ☐ (C) sub3 ☐ (D) sub2
☐ (E) Báo lỗi không tìm thấy a

5. Nếu đoạn code trên được viết trên ngôn ngữ dùng qui tắc tầm vực động (dynamic-scope rule), và giả sử chuỗi gọi là $\text{main} \rightarrow \text{sub1} \rightarrow \text{sub2} \rightarrow \text{sub3}$, tham khảo đến a trong sub3 **khi thực thi** ứng với kết hợp của a trong bản hoạt động của

- ☐ (A) main ☐ (B) Báo lỗi không tìm thấy a ☒ (C) sub2 ☐ (D) sub1
☐ (E) sub3

6. Trình định thời (scheduled subprograms) thường được dùng trong

- ☐ (A) Lập trình hướng đối tượng (Object-Oriented Programming)
☐ (B) Lập trình song song (Parallel Programming)
☐ (C) Lập trình hướng sự kiện (Event-driven Programming)
☐ (D) Lập trình hàm (Functional Programming)
☒ (E) Lập trình thời gian thực (Real-time Programming)

7. Liệt kê theo thứ tự chiều dài tăng dần ít nhất 5 chuỗi ngắn nhất (nếu có nhiều hơn 5 chuỗi) của ngôn ngữ được mô tả bởi biểu thức chính quy (regular expression) sau:
 $\text{ab}^*(\text{a|b})^? \text{a}$

- ☐ (A) ϵ , a, aa, aba, abaa ☐ (B) aa, aba, aaa, abaa, aaba ☐ (C) a, aa, aba, aaa, aaba
☒ (D) aa, aba, aaa, abaa, abba ☐ (E) aa, aba, aaa, abaa, aaaa

8. Giả sử chương trình có một lỗi văn phạm nằm sau lệnh **print**. Khi em nhấn nút Run (để dịch và chạy chương trình) trên trình soạn thảo, kết quả được in ra bởi lệnh **print** trên trước khi lỗi văn phạm được báo. Hỏi chế độ dịch của trình soạn thảo là gì?

- ☐ (A) trình biên dịch (compiler) ☐ (B) trình liên kết (link editor)
☐ (C) trình hợp ngữ (assembler) ☒ (D) trình thông dịch(interpreter)
☐ (E) trình biên dịch động(just-in-time compiler)

Đoạn mã sau được dùng trong các câu 9– 10:

```
int *p = new int;
void foo(int * r) {
    delete r;
}
foo(p); //1
*p = 2; //2
```

9. Hiện tượng gì xảy ra khi p được truyền cho r ở phát biểu //1 trong đoạn mã trên:

- ☒ (A) Bí danh (alias) ☐ (B) Con trỏ chưa khai báo (undeclared pointer) ☐ (C) Tham chiếu treo (dangling reference)
☐ (D) Khai báo trùng tên (redeclared) ☐ (E) Rác (garbage)

14. Cho giá trị ban đầu của biến **c** là 3, cho biết những giá trị **có thể có** của biến **a** sau khi thực hiện phép gán sau trên ngôn ngữ lập trình C?
 $a = c * (c = 5);$

(A) 15 (B) 15, 25, 9 (C) 9 (D) 25 **(E) 15, 25**

15. Viết lại biểu thức trung tố (infix) sau sang dạng hậu tố (postfix) Polish? Giả sử là tất cả các phép toán đều có 2 toán hạng và có độ ưu tiên và tính kết hợp như trong ngôn ngữ lập trình C.
 $a + b * c - d * e + f$

(A) $a b + c * d - e * f +$ (B) $a + b c * d e * - f +$ (C) $a b c * + d e * + f -$
 (D) $a b c * + d - e f * +$ **(E) $a b c * + d e * - f +$**

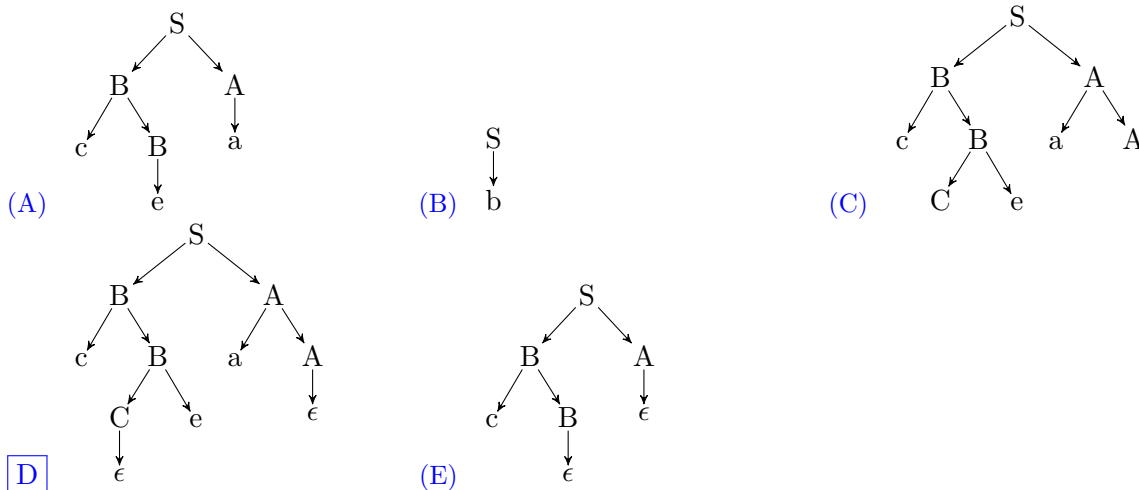
Cho văn phạm sau dùng cho các câu 16-17:

$S \rightarrow b \mid B A$
 $A \rightarrow a A \mid \epsilon$
 $B \rightarrow C e \mid c B$
 $C \rightarrow d C \mid \epsilon$

16. Chuỗi nào KHÔNG thuộc ngôn ngữ được mô tả bởi văn phạm trên?

(A) b (B) e (C) cccddddeaa (D) eaaa **(E) cccaaa**

17. Chọn cây phân tích cú pháp (parse tree) cho chuỗi nhập: **cea**



18. Cho biết mã Jasmin của biểu thức viết bằng ngôn ngữ C như sau:

$a * b - 2$

với a và b là 2 biến nguyên có chỉ số lần lượt là 0 và 1.

(A) $\begin{matrix} \text{iload_0} \\ \text{iload_1} \\ \text{iconst_2} \\ \text{imul} \\ \text{isub} \\ \dots \end{matrix}$ **(B) $\begin{matrix} \text{iload_0} \\ \text{iload_1} \\ \text{imul} \\ \text{iconst_2} \\ \text{isub} \\ \dots \end{matrix}$** (C) $\begin{matrix} \text{iload_0} \\ \text{iload_1} \\ \text{iconst_2} \\ \text{isub} \\ \text{imul} \\ \dots \end{matrix}$ (D) $\begin{matrix} \text{iload_0} \\ \text{imul} \\ \text{iload_1} \\ \text{isub} \\ \text{iconst_2} \\ \dots \end{matrix}$ (E) $\begin{matrix} \text{imul} \\ \text{load_0} \\ \text{iload_1} \\ \text{isub} \\ \text{iconst_2} \\ \dots \end{matrix}$



19. Cho biết kích thước của đối tượng dữ liệu **x** được khai báo như sau:
x: set of 1..16

- ☐ (A) 2 bytes ☐ (B) 4 bytes ☐ (C) 1 byte ☐ (D) 4 bits
☐ (E) 16 bytes

20. Để thực hiện tìm kiếm một chuỗi (String) trên nhiều loại danh sách khác nhau, hãy chọn giải pháp thích hợp nhất để điền vào chỗ trống trong định nghĩa hàm sau:

```
def lookup[T](x:String, lst:List[T], f:T=>String):Option[T] =  
  lst match {  
    case List() => None  
    case h::t => _____  
  }
```

- ☐ (A) if (f(h) == x) Some(h) else lookup(x,t,f)
☐ (B) if (h == x) Some(h) else lookup(x,t,f)
☐ (C) if (f(h) == x) Some(f(h)) else lookup(x,t,f)
☐ (D) if (h == f(x) Some(h) else lookup(x,t,f)
☐ (E) if (h == f(x)) Some(f(h)) else lookup(x,t,f)

21. Chọn phát biểu ĐÚNG trong các phát biểu về dãy (array) sau:

- ☐ (A) Truy xuất một phần tử bên ngoài một dãy C không được phép, sẽ lập tức gây ra lỗi
☐ (B) Trên C, kích thước của dãy có thể tăng thêm sau khi dãy đã được tạo ra
☐ (C) Trên Java, kích thước của một dãy KHÔNG được lưu trữ để kiểm tra động
☐ (D) Trên C và C++, kích thước của một dãy được lưu trữ để kiểm tra động
☐ (E) Truy xuất một phần tử bên ngoài một dãy Java không được phép, sẽ lập tức gây ra lỗi

22. Cơ chế gọi chương trình con nào mà điều khiển có thể chuyển vào một vị trí ở giữa chương trình con được gọi thay vì chuyển vào đầu chương trình con được gọi?

- ☐ (A) Gọi trở về đơn giản (Simple Call - Return)
☐ (B) Trình định thời (Scheduled Subprogram)
☐ (C) Biến cố - Xử lý biến cố (Exception)
☐ (D) Gọi đệ qui (Recursive call)
☐ (E) Trình cộng hành (Coroutine)

Phần hướng dẫn này áp dụng cho các câu 23-27

Hãy viết thêm các đoạn mã cần thiết vào phương thức dưới đây để thực hiện sinh mã cho phát biểu *repeat*. Nhắc lại phát biểu *repeat* kết thúc quá trình lặp khi điều kiện của phát biểu trở thành **true**. AST của phát biểu **repeat** được định nghĩa như sau:

```
case class Repeat(val stmts:List[Stmt],val exp:Expr) extends Stmt
```

```
override def visitRepeat(ast: Repeat, o:Context)= {  
  val ctxt = o.asInstanceOf[SubBody]  
  ctxt.frame.enterLoop();  
  val labelStart = ctxt.frame.getNewLabel()  
  val labelBreak = ctxt.frame.getBreakLabel()  
  val labelCont = ctxt.frame.getContinueLabel()  
  val str1 = //1  
  // sinh mã cho từng phát biểu trong thân của repeat
```

```
val str2 = //2
val str3 = //3
// sinh mã cho biểu thức điều kiện
val str4 = visit(ast.exp,...)
val str5 = //4
val str6 = //5
frame().exitLoop();
str1 + str2 + str3 + str4 + str5 + str6
}
```

23. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //1 trong đoạn mã trên

- ☐ (A) ctxt.emit.emitLABEL(labelStart) ☐ (B) ""
☐ (C) ctxt.emit.emitLABEL(labelCont) ☐ (D) ctxt.emit.emitLABEL(labelBreak)
☐ (E) ctxt.emit.emitGOTO(labelStart)

24. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //2 trong đoạn mã trên

- ☐ (A) ast.stmts.map(x=>visit(x,o))
☐ (B) visit(ast.stmts,o)
☐ (C) ast.stmts.foldLeft(")((x,y)=>x + visit(y,o))
☐ (D) ast.stmts.filter(x=>visit(x,o))
☐ (E) ast.stmts.foldLeft(")((x,y)=>y + visit(x,o))

25. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //3 trong đoạn mã trên

- ☐ (A) ctxt.emit.emitIFFALSE(labelStart) ☐ (B) ""
☐ (C) ctxt.emit.emitLABEL(labelCont) ☐ (D) ctxt.emit.emitLABEL(labelBreak)
☐ (E) ctxt.emit.emitGOTO(labelStart)

26. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //4 trong đoạn mã trên

- ☐ (A) ctxt.emit.emitIFFALSE(labelStart) ☐ (B) ctxt.emit.emitGOTO(labelBreak)
☐ (C) ctxt.emit.emitLABEL(labelCont) ☐ (D) ctxt.emit.emitLABEL(labelBreak)
☐ (E) ctxt.emit.emitGOTO(labelStart)

27. Đoạn mã nào trong số các đoạn mã dưới đây cần phải xuất hiện ở vị trí //5 trong đoạn mã trên

- ☐ (A) ctxt.emit.emitIFFALSE(labelStart)) ☐ (B) ctxt.emit.emitGOTO(labelBreak))
☐ (C) ctxt.emit.emitLABEL(labelCont)) ☐ (D) ctxt.emit.emitLABEL(labelBreak))
☐ (E) ctxt.emit.emitGOTO(labelStart))

28. Cho $X = \{a,b\}$. Chọn biểu thức chính quy (regular expression) mô tả ngôn ngữ chứa bất kỳ chuỗi nào được tạo bởi các ký tự trên tập X nhưng KHÔNG chứa chuỗi có 2 ký tự a liên tiếp.

- ☐ (A) $b^*a?(bb^*a)^*b^*$ ☐ (B) $a|(abb^*)^*$ ☐ (C) $b^*(abb^*)^*$
☐ (D) $(b^*ab^*)^*$ ☐ (E) $b^*ab^*ab^*$



Đoạn code sau dùng cho các câu 29–31

```
int p;  
int* foo(int x) {  
    static int q;  
    int *s = new int;  
    switch (x) {  
        case 1: return &p;  
        case 2: return &q;  
        case 3: return &x;  
        case 4: return s;  
        default: return foo(x-1);  
    }  
}
```

29. Phát biểu nào sẽ gây ra lỗi tham chiếu treo (dangling reference) khi thực thi

- (A) return s (B) return &p ☒ (C) return &x (D) return &q
(E) Không phát biểu nào gây ra lỗi trên khi thực thi

30. Trong đoạn mã trên, đối tượng dữ liệu nào có thể trở thành rác (garbage)

- ☒ (A) trỏ đến bởi s (B) p (C) x (D) q
(E) Không có đối tượng nào có thể trở thành rác

31. Khi hàm **foo** được gọi đệ quy (recursive) thì các bản hoạt động của **foo** dùng chung những đối tượng dữ liệu nào?

- (A) p, q, x và s (B) q và s ☒ (C) p và q (D) p
(E) x, q và s

32. Cho khai báo sau được viết trên ngôn ngữ lập trình Ada:

```
type Shape is ( Circle , Triangle , Rectangle );  
type Colors is ( Red , Green , Blue );  
type Figure (Form: Shape) is record  
    Filled: Boolean;  
    Color: Colors;  
    case Form is  
        when Circle => Diameter: Float;  
        when Triangle =>  
            Leftside , Rightside: Integer;  
            Angle: Float;  
        when Rectangle => Side1 , Side2: Integer;  
    end case;  
end record;
```

Cho kích thước của các kiểu *Integer*, *Float*, *Boolean* và *Enumeration* lần lượt là 2, 4, 1, và 2, và kích thước cho phần mô tả kiểu (type description) là 0. Cho biết kích thước của một đối tượng kiểu **Figure**?

- (A) 15 (B) 21 (C) 23 ☒ (D) 13 (E) Khác

Đoạn mã sau được dùng cho các câu 33–38.

Cho một đoạn chương trình được viết trên một ngôn ngữ tựa C như sau:



```
int A[5] = {1,3,5,7,9}; // index of A starts from 0
int j = 0;
int n = 5;
int sumAndIncrease(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 = sumAndIncrease(A[j], j);
    printf("a = %i\n", s); //1
    printf(" %i %i %i %i %i\n", A[0], A[1], A[2], A[3], A[4]); //2
}
```

33. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 25 ☒ (B) 5 (C) 15 (D) 10 (E) Khác

34. Nếu a và i được truyền bằng **trị-kết quả (passed by value-result)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 5 3 5 7 9 (B) 1 3 5 7 9 ☒ (C) 6 3 5 7 9 (D) 2 4 6 8 10 (E) Khác

35. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- (A) 25 (B) 5 (C) 15 (D) 10 ☒ (E) Khác

36. Nếu a và i được truyền bằng **tham khảo (passed by reference)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 5 3 5 7 9 (B) 1 3 5 7 9 (C) 6 3 5 7 9 ☒ (D) 2 4 6 8 10 (E) Khác

37. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến s được in ra ở phát biểu //1 là bao nhiêu?

- ☒ (A) 25 (B) 5 (C) 20 (D) 15 (E) Khác

38. Nếu a và i được truyền bằng **tên (passed by name)**, giá trị của biến A được in ra ở phát biểu //2 là bao nhiêu?

- (A) 5 3 5 7 9 (B) 1 3 5 7 9 (C) 6 3 5 7 9 ☒ (D) 2 4 6 8 10 (E) Khác



39. Cho lớp(class) A là lớp cha (superclass) của lớp B và B là lớp cha của hai lớp C và D. Trong lớp A, có khai báo phương thức thực thể (instance method) **foo** và phương thức này bị ghi đè(override) trên tất cả các lớp B, C và D. Cho biến x được khai báo như sau:
B x;

Qui ước viết *func_T* là phương thức func được khai báo trong lớp T, cho biết tập của phương thức đích của cuộc gọi sau: **x.foo();**

- (A) {foo_A,foo_B,foo_C,foo_D} (B) {foo_C,foo_D} (C) {foo_A,foo_B}
(D) {foo_B} (E) {foo_B,foo_C,foo_D}

40. Cơ chế gọi chương trình nào là cơ chế gọi chương trình con cơ bản của Lập trình hướng sự kiện (Event-driven Programming)

- (A) Gọi trở về đơn giản (Simple Call - Return)
(B) Biến cố - Xử lý biến cố (Exception)
(C) Trình cộng hành (Coroutine)
(D) Gọi đệ qui (Recursive call)
(E) Trình định thời (Scheduled Subprogram)

41. Một biến trên ngôn ngữ Javascript có thể nhận những giá trị thuộc các kiểu khác nhau. Ví dụ:

```
var x = "def";  
x = 1;
```

Ràng buộc giữa một biến và kiểu của nó trên ngôn ngữ Javascript xảy ra trong thời gian

- (A) Chạy (running) (B) Hiện thực (implementation)
(C) Dịch (compiling) (D) Lập trình (programming)
(E) Khởi động (loading)

42. Gọi append là hàm dùng để nối 2 danh sách thành 1 danh sách. Ví dụ append([1,2,3],[4,5,6]) sẽ có kết quả là [1,2,3,4,5,6]. Hãy hiện thực hàm append(a:List[Int],b:List[Int]) dùng hàm bậc cao (high-order function)?

- (A) a.foldLeft(b)((x,y)=>x::y) (B) a.map(b)((x,y)=>x::y)
(C) b.foldLeft(a)((x,y)=>x::y) (D) a.foldRight(b)((x,y)=>x::y)
(E) b.foldRight(a)((x,y)=>x::y)

Phần trình bày sau dùng trong các câu hỏi 43– 44:

Cho mã giả của phát biểu *for var = expr1 to expr2 do body* như sau:

```
start:  var = expr1  
        etemp = expr2  
loop:   if (var >= etemp) goto out  
        body  
label1: var++  
label2: goto loop  
out:
```

43. Nếu trong thân *body* có thực thi lệnh *continue* thì điều khiển sẽ chuyển đến thực thi phát biểu có nhãn là:

- (A) start (B) out (C) label1 (D) loop (E) label2

47. Kiểu *union* của ngôn ngữ C++ cho phép ghi vào ở thành phần này nhưng đọc lên ở thành phần khác như trong ví dụ sau:

```
union {  
    int songuyen;  
    char kytu[2];  
} x;  
x.songuyen = 12;  
cout << x.kytu[0] << " " << x.kytu[1] << endl;
```

Đặc tính này của ngôn ngữ C++ gây ảnh hưởng XẤU NHẤT đối với

- | | |
|--------------------------------|------------------------------------|
| (A) tính đơn giản (simplicity) | (B) tính trực giao (orthogonality) |
| (C) tính tin cậy (reliability) | (D) tính dễ viết (writability) |
| (E) chi phí (cost) | |

II. Phần câu hỏi tự luận: (dành cho tất cả sinh viên) (2 điểm)

48. 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

Lời giải. $(+ a (- (* b c d) e f))$

Nếu viết đúng biểu thức dạng tiền tố Cambridge Polish: 1

Nếu viết đúng và có số () ít nhất: 1

49. Hãy nêu những giải pháp đã được sử dụng để tránh tham chiếu treo (dangling reference) trong trường hợp trả về hàm như sau:

```
void->int F() {  
    int x = 1;  
    int g() {  
        return x + 1;  
    }  
    return g ;  
}  
void->int gg = F();  
int z = gg();
```



50. Trên những ngôn ngữ lập trình cho phép rút ngắn tính toán (short-circuit evaluation) biểu thức luận lý, biểu thức sau sẽ không gây ra lỗi khi thực thi.

$((a \neq 0) \ \&\& \ ((b / a) > 5))$

Hãy dùng **if then else** để mô phỏng quá trình tính toán của biểu thức luận lý trên?

Lời giải. if $((a \neq 0)$ then if $((b / a) > 8)$ then ...

Giải pháp đề xuất đúng: 2

III. Phần bài tập lớn: (dành cho tất cả sinh viên)

51. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện kiểm tra kiểu. Hàm `visitBlock` sẽ thực hiện chức năng xây dựng bảng danh hiệu (symbol table) và truyền bảng này khi visit các phát biểu trong khối. Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm. Nhắc lại AST của Block được khai báo như sau:

case class Block(val decls:List[Decl],val stmts:List[Stmt]) extends Stmt

52. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện sinh mã cho một phát biểu If. Nhắc lại AST của If được khai báo như sau:

case class If(val expr:Expr, val thenStmt:Stmt, val elseStmt:Option[Stmt]) extends Stmt

Yêu cầu giải thích rõ về Context cụ thể được sử dụng trong hàm và ý tưởng mã của hàm.

IV. Phần KSTN: (dành cho sinh viên KSTN)

53. Viết hàm `visitIf(ast:If,o:Context)` để thực hiện kiểm tra lỗi FunctionNotReTurn? Yêu cầu giải thích ý tưởng thực hiện thể hiện qua mã của hàm?

54. Viết hàm `visitBlock(ast:Block,o:Context)` để thực hiện sinh mã cho một khối (block)? Yêu cầu giải thích ý tưởng thực hiện sinh mã cho các khai báo array trong một khối?



Question Distribution

Content	Level 1	Level 2	Level 3	Level 4
Introduction		1		
Lexical	1	4	5	
Syntax	2	4	7	
AST		2		1
Total	3	11	12	1

HẾT

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