프로그래밍언어론 기말고사(2017년 2학기)

담당교수 : 박두순

※**<주의> 모든 문제는 정확하게 설명되어야만 점수를 받을 수 있습니다.**

1. 다음에 대해서 간단하게 정의하시오. **(각 5점)**

a) 현수 else(dangling else)　　b) 주석 파스 트리(annotated parse tree)

(풀이)

a) 중첩된 if 문장에서 else가 어떤 if 문에 걸려있느냐 하는 것(5점)

b) 각 노드에서 속성 값을 주석으로 보여주는 파스 트리(5점)

2. 언어구현기법은 번역기법, 순수해석기법(인터프리터 기법), 혼합형 기법 등이 있다. 3가지 방법에 대해서 정의, 장단점, 언어에 대해서 설명하시오.**(10점)**

(풀이) 개념(4점), 장단점(4점), 언어(2점)

	번역기법	인터프리터기법	혼합형
입력	고급언어	고급언어	컴파일러와 인터프리터를 같이 사용하는 기법으로 고급 언어를 인터프리테이션이 가능하도록 중간 언어로 번역
출력	목적 프로그램	실행결과	
장점	반복처리의 경우 실행시간 단축	사용자의 융통성, 쉽게 구현 가능	원시 프로그램이 한번만 번역되기 때문에 인터프리터 기법보다 빠르다.
단점	큰 기억장치요구, 사용자의 융통성 결여	실행시간이 많이 걸림	인터프리터 기법보다 융통성이 없다.
언어	포트란, 파스칼, c	APL, LISP	JAVA, PERL

3. 다음 **Java 문장**에서 토큰, 패턴, 렉심을 설명하시오.**(10점)**

index = 2 * count + 17 ;

(풀이) 토큰, 렉심, 패턴 각 3점, 모두 맞으면 10점

토큰	패턴	렉심
식별자	영문자 소문자(a-z)와 대문자(A-Z), 밑줄 문자(_), 달러($)로 시작하고 두 번째 문자부터는 영문자 소문자(a-z)와 대문자(A-Z), 숫자(0-9), 밑줄 문자(_), 달러($)로 구성된다.	index
치환 연산자	=	=
정수형 상수	숫자(0-9)로 시작하고 두 번째 문자부터도 숫자(0-9)로 구성	2
곱셈 연산자	*	*
식별자	영문자 소문자(a-z)와 대문자(A-Z), 밑줄 문자(_), 달러($)로 시작하고 두 번째 문자부터는 영문자 소문자(a-z)와 대문자(A-Z), 숫자(0-9), 밑줄 문자(_), 달러($)로 구성된다.	count
덧셈 연산자	+	+
정수형 상수	숫자(0-9)로 시작하고 두 번째 문자부터도 숫자(0-9)로 구성	17
구분자	;	;

4. 다음 문법 G에서 문장 A = A * (B + (C)) 가 맞는 문장인지 좌단 유도를 통해 보이고, 좌단 유도에 의해 생성되는 파스 트리를 그리고, 문장과 문장형태도 설명하시오. **(25점)**

P : <assign> → <id> = <expr>

<expr> → <id> + <expr> | <id> - <expr> | <id> * <expr> | (<expr>) | <id>

<id> → A | B | C | D

(풀이)

문장 A = A * (B + (C)) 에 대해서 좌단유도를 하면

<assign> ⇒ <id> = <expr>

⇒ A = <expr>

⇒ A = <id> * <expr>

⇒ A = A * <expr>

⇒ A = A * (<expr>)

⇒ A = A * (<id> + <expr>)

⇒ A = A * (B + <expr>)

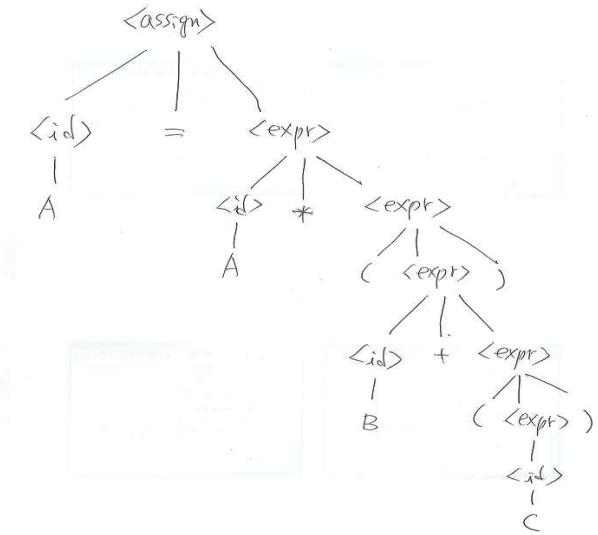
⇒ A = A * (B + (<expr>))

⇒ A = A * (B + (<id>))

⇒ A = A * (B + (C))

문장 A = A * (B + (C))는 맞는 문장이다. (10점)

좌단 유도에 의한 파스 트리를 그리면 다음과 같다.



(10점)

문장 : A = A * (B + (C))(1점)

문장형태 : <assign>, <id> = <expr>, A = <expr>, A = <id> * <expr>, A = A * <expr>,
A = A * (<expr>), A = A * (<id> + <expr>), A = A * (B + <expr>),
A = A * (B + (<expr>)), A = A * (B + (<id>)), A = A * (B + (C)) (4점)

5. 다음과 같은 문법이 모호한 문법인지 아닌지를 보이고, 만일 모호한 문법이라면 다음의 규칙에 맞게 모호하지 않은 문법으로 바꾸시오.**(20점)**

< S > → < A >

< A > → < A > * < A > | < id >

< id > → x | y | z

단, * 는 연산자이고, *는 왼쪽 결합법칙을 따른다.

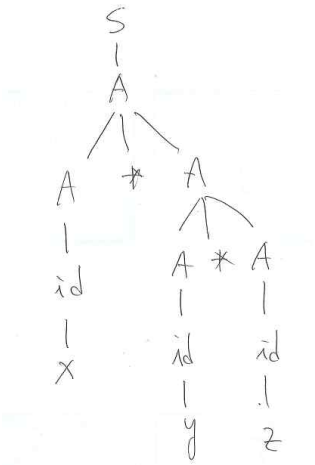
(풀이) 식을 뭐를 주느냐에 따라 설명이 모두 다름

식 x * y * z에 대해서 모호한 문법인지 아닌지를 살펴보자. 식 x * y * z에 대해서 서로 다른 두 개 이상의 파스트리가 만들어 지면 모호한 문법이다. 이를 확인해보자. 이를 위해 좌단유도를 통해 문법으로부터 식 x * y * z를 유도하자.

첫 번째로

<S> ⇒ <A>
⇒ <A> * <A>
⇒ <id> * <A>
⇒ x * <A>
⇒ x * <A> * <A>
⇒ x * <id> * <A>
⇒ x * y * <A>
⇒ x * y * <id>
⇒ x * y * z

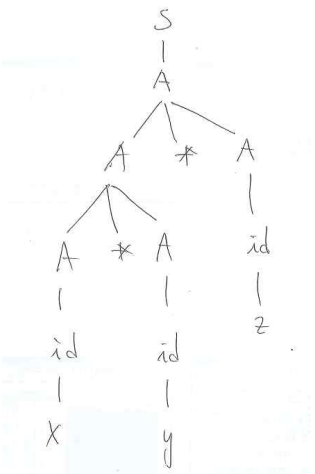
이를 파스트리로 표현하면 다음과 같다.



두 번째로

<S> ⇒ <A>
⇒ <A> * <A>
⇒ <A> * <A> * <A>
⇒ <id> * <A> * <A>
⇒ x * <A> * <A>
⇒ x * <id> * <A>
⇒ x * y * <A>
⇒ x * y * <id>
⇒ x * y * z

이를 파스트리로 표현하면 다음과 같다.



이와 같이 식 x * y * z 에 대해 서로 다른 두개의 파스트리가 만들어지므로 주어진 문법은 모호한 문법이다. (10점)

이를 모호하지 않은 문법으로 변환하자. *를 왼쪽 결합법칙을 따르도록 모호하지 않은 문법으로 변환해보자.

우선 가장 기초적인 피연산자를 먼저 처리하고 이 생성 규칙이 생성 규칙 중에서 가장 아래에 온다. 가장 기초적인 피연산자를 <F>라 하면 다음과 같다.

< F > → x | y | z

다음으로는 * 연산자에 대해서 왼쪽 결합 법칙을 취하므로 다음 생성 규칙과 같이 재귀적으로 구성

< T > ::= < T > * < F > | < F >

나머지도 그대로 처리하면 된다.

< S > → < T >

< T > → < T > * < F > | < F >

< F > → x | y | z (10점)

(식별자 이름을 다른 이름으로 사용하거나 id 사용하거나 모두 맞음. 시작 기호는 S이어야 함.)

6. 다음의 확장된 문법에 대한 SLR 파싱 표가 다음과 같을 때 입력 id+id*id에 대해서 파싱하면서 파스트리도 설명하시오.**(20점)**

(0) S → E (1) E → E + T (2) E → T (3) T → T * F (4) T → F (5) F → (E) (6) F → id

상태	구문분석기의 행동						GOTO 함수		
	id	+	*	(	)	$	E	T	F
0	S5			S4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

(풀이)

스택	입력	행동
0	id+id*id$	shift 5
5id0	+id*id$	reduce 6
F0	+id*id$	
3F0	+id*id$	reduce 4
T0	+id*id$	
2T0	+id*id$	reduce 2
E0	+id*id$	
1E0	+id*id$	shift 6
6+1E0	id*id$	shift 5
5id6+1E0	*id$	reduce 6
F6+1E0	*id$	
3F6+1E0	*id$	reduce 4
T6+1E0	*id$	
9T6+1E0	*id$	shift 7
7*9T6+1E0	id$	shift 5
5id7*9T6+1E0	$	reduce 6
F7*9T6+1E0	$	
10F7*9T6+1E0	$	reduce 3
T6+1E0	$	
9T6+1E0	$	reduce 1
E0	$	
1E0	$	accept

(18점)

그러므로 주어진 문장은 주어진 문법에 맞는 문장이다. (2점)

7. 바인딩과 바인딩 시간에 대해서 간단하게 설명하고 C 언어의 치환문 count = count + 20 ; 에서 발생될 수 있는 바인딩 중에서 다음 물음에 답하시오. 또한, 바인딩에 따른 언어 두 개 씩을 서술하시오. **(20점)**

①count에 대한 가능한 자료형의 집합 ②count의 자료형 ③count의 가능한 값들의 집합
④count의 값 ⑤혼합형 연산을 허용하는 언어라면 혼합형 연산에 대해 어떻게 연산할 것인지?
⑥연산자 '+'에 대한 가능한 의미의 집합 ⑦연산자 '+'의 의미 ⑧리터럴 20의 내부 표현
⑨리터럴 20이 10진수의 수라는 것

(풀이)

바인딩(binding)은 프로그램의 기본단위에 이 기본단위가 택할 수 있는 여러 가지 속성(attribute)중에서 일부를 선정하여 결정해주는 행위(변수의 타입, 변수의 storage, 변수의 값 등) (5점)

바인딩 시간(binding time)은 바인딩이 일어나는 시간 (2점)

C 언어이므로 발생 가능한 바인딩은 다음과 같다. (각 1점)

①count에 대한 가능한 자료형의 집합 - 언어설계 시간
②count의 자료형 – 컴파일 시간
③count의 가능한 값들의 집합 - 언어구현시간
④count의 값 – 실행시간
⑤혼합형 연산을 허용하는 언어라면 혼합형 연산에 대해 어떻게 연산할 것인지? - 언어설계시간
⑥연산자 '+'에 대한 가능한 의미의 집합 – 컴파일 시간
⑦연산자 '+'의 의미 – 언어설계시간
⑧리터럴 20의 내부 표현 - 언어구현시간
⑨리터럴 20이 10진수의 수라는 것 - 언어설계시간

번역 시간에 맞는 언어는 FORTRAN, ALGOL, COBOL
실행 시간에 맞는 언어는 SNOBOL 4, APL, LISP (2개 이상 맞을 때 4점)

8. 다음과 같은 ALGOL 형태의 프로그램에서 변수 i와 j에 대해서 정적 영역(static scope), 동적 영역(dynamic scope), 영역 구멍(hole-in-scope)에 대해서 설명하시오.**(20점)**

```
1. A : begin integer i, j; real x, y;
2. B : procedure test(integer a, b)
3.          begin boolean i;
4.              ……
5.              x: = i * j + y ;
6.              ……
7.          end B;
8.          ……
9.   C : begin integer x, y; real i, j
10.             ……
11. D : begin boolean j;
12.             ……
13.          call test(x, y);
14.             ……
15.       end D;
16.             ……
17.       end C;
18.       ……
19.       end A;
```

(풀이) 정적 모두 맞으면 8점, 동적 모두 맞으면 8점, 두다 맞으면 20점

정적 영역은 다음과 같다. (각 항목 8점)

변수 **선언된 자료형**	**i**	**j**
integer	1, 2, 8, 18-19	1-8, 18-19
real	9-17	9-10, 16-17
boolean	3-7	11-15
영역구멍	3~7, 9-17	9~17

동적 영역은 다음과 같다. (각 항목 8점)

변수 **선언된 자료형**	**i**	**j**
integer	1, 8, 18-19	1, 8, 18-19
real	2, 9-17	9-10, 16-17
boolean	3-7	11-15, 2-7
영역구멍	9~17, 2-7	9~17

9. 다음과 같이 C 언어에서 2차원 배열이 선언되어 있을 때 aa[4][2]에 대해 주소를 계산하시오. 단, 정수형은 4바이트를 할당한다. **(15점)**

```
int aa[7][9]
```

(풀이)

C 언어는 행우선 방법으로 저장되므로 2차원 배열의 경우 aa[i][j]의 상대 주소는 다음과 같다.

i는 4, j는 2, low1은 행의 하한 값이므로 0, low2는 열의 하한 값이므로 0, n2는 열의 크기이므로 9, w는 4이므로

base + (((i - low1) × n2) + (j - low2 + 1)) × w

= base + (((4 - 0) × 9) + (2 - 0 + 1)) × 4
= base + (36 + 3) × 4
= base + 156 (15점)

10. 다음 ALGOL형태의 프로그램에서 call by reference, call by value, call by name 기법으로 매개변수 전달을 한 경우 출력되는 값을 구하는 과정을 상세하게 설명하시오.**(15점)**

```
BEGIN INTEGER A, B;
    PROCEDURE F(X, Y, Z); INTEGER X, Y, Z;
      BEGIN Y := Y+5;
            Z := Z+X
      END F;
   A := 10; B := 20; F(A+B, A, A);
   PRINT A
END
```

(풀이)

(1) 참조호출의 경우 :

처음에 A = 10, B = 20이다. 프로시저 F가 호출되면 F(A+B, A, A)에서 addr(A+B) = addr(X), addr(A) = addr(Y), addr(A) = addr(Z)이다. 여기서 addr(A+B) = addr(X)는 처음의 A+B의 값인 30을 X의 주소에 저장한다는 의미이다. 또한, A, Y Z의 주소는 동일하다. Y := Y + 5에서 Y = 10+5 = 15, Z := Z + X에서 Z = 15 + 30 = 45이다. PRINT A에서 A의 주소는 Z의 주소와 같으므로 45를 출력한다. (5점)

(2) 값 호출의 경우 :

처음에 A = 10, B = 20이다. 프로시저 F가 호출되면 F(A+B, A, A)에서 X, Y, Z에 대한 기억 장소를 별도로 할당한 다음 X = 30, Y = 10, Z = 10을 저장한다. Y := Y + 5에서 Y = 10+5 = 15, Z := Z + X에서 Z = 10 + 30 = 40이다. 그런데 프로시저가 리턴 되면서 X, Y, Z에 대한 기억장소가 삭제된다. PRINT A에서 10을 출력한다. (5점)

(3) 이름 호출의 경우 :

처음에 A = 10, B = 20이다. 프로시저 F가 호출되면 F(A+B, A, A)가 호출된 다음 Y := Y + 5에서 A = A+5 = 10+5 = 15, Z := Z + X에서 A = A + (A + B) = 15 + 15 + 20 = 50 PRINT A에서 50을 출력한다.(5점)

GOOD LUCK !!