CENG 442: Programming Language Concepts **16.04.2014**

Midterm

1. (10 points) Consider the traditional grammar for arithmetic expressions that follows:

E → E + T | E - T | T
T → T * F | T / F | F
F → (E) | id

Draw a derivation (parse) tree for the arithmetic expression (id + id) * id.

2. (10 points) Convert the following BNF to EBNF. Use each of {}, [] and () at least once.

S → AS | SB | ab
A → aB | bB
B → bA | b

3. (10 points) Consider the grammar that follows:

<stmt> → if <expr> then <stmt> | if <expr> then <stmt> else <stmt> | a=a+1
<expr> → a=b | a > b

Prove that this grammar is ambigious.

4. (10 points) Write an equivalent unambiguous grammer for the grammer at question 3.

5. (15 points) The code below is written in C++. In C++ language, when we execute the statement "`int x = 0;`", x becomes a stack-dynamic variable. And when we execute the statement "`static int y = 0;`", y becomes a static variable. What is the output of the program below?

```
int sum(int a){

      int x = 0;
      static int y = 0;

      if(a==1){
            return a;
      }
      x = x + sum(a-1);
      y = y + sum(a-1);
      cout << x << " " << y << endl;
      return x+y;

}

int main()
{
      sum(5);
}
```

6. (10 points) The code below is written in C++. In C++ language, when we execute the statement a=b; , a and b do not become alias. When we execute the statement, int &c=d; , c and d become alias. When we execute the statements, int *e; e = &f; , *e and f become alias. What is the output when we execute the code?

```
int a = 1;
int b = 2;
int &c=a;
int *ptr;
ptr = &c;
cout<< a << " " << b << " " << c <<" " << *ptr <<endl;
c = b;
b = c;
ptr = &b;
c = 3;
cout<< a << " " << b << " " << c <<" " << *ptr <<endl;
a = 4;
*ptr = 5;
cout<< a << " " << b << " " << c <<" " << *ptr <<endl;
*ptr = 6;
ptr = &a;
cout<< a << " " << b << " " << c <<" " << *ptr <<endl;
```

7. (10 points) The code below is written in C++. x is a global variable. In C++ language, we can use scope operator " :: " to reach a global variable. What is the output?

```
int x = 2000;
void display1(){
     int x = 1;
     x = x + 1;
     ::x = ::x + 10;
     cout << "1: " << x << " " << "1: " << ::x <<endl;
}
void display2(){
     x = x + 100;
     cout << "2: " << x << endl;
}
int main()
{
     display1();
     display1();
     display2();
     display1();
     display2();
     display1();

     cout << "Main: " << x << endl;
}
```

8. (10 points) Consider the code below:

```
function fun1(){
     function fun2(){
          int x=3
          print (x, y)
          fun4()
     }
     function fun3(){
          int y=4
          print (x, y)
          fun2()
     }
     function fun4(){
          print (x, y)
     }
     int x=1
     int y=2
     print (x, y)
     fun2()
     fun3()
}
```

When we call function fun1, what is the output if language uses static scooping? When we call function fun1, what is the output if language uses dynamic scooping?

9. (15 points) Consider the traditional grammar for arithmetic expressions that follows:

1. $E \rightarrow E + T$
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

The figure below shows the LR parsing table for this grammar. Write a trace of a parse of the string id * (id + id) using the LR parsing algorithm and the parsing table below.

	Action						Goto		
State	id	+	*	(	)	$	E	T	F
0	S5			S4			1	2	3
1		S6				accept			
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			