Southeast University Examination Paper (A)

Course Name Principles of Compiling Examination Term 08-09-2 Score

Related Major Computer Science & Technology Examination Form Close test Test Duration 150 Mins

There are 8 problems in this paper. You can write the answers in English or Chinese on the attached paper sheets.

1. Please construct context-free grammars **with ε-free productions** for the following language (10%).

   {ω| ω∈(a,b,c)* and the numbers of a's and b's and c's occurred in ω are **even**, and ω **starts with b , ends with a or c**}

2. Please construct a **DFA with minimum states** for the following regular expression. (10%)

   (a|(a|(a|b*))*)*(a|b*)

3. Please **eliminate the left recursions (if there are)** and **extract maximum common left factors (if there are)** from the following context free grammar, and then decide **the resulted grammar** is whether a LL(1) grammar by **constructing the related LL(1) parsing table.**(15%)

   P→b S d

Surname_____

Student No._____

Please obey the rules of examination. If you violate the rules, your answer sheets will be invalid

S→S ; A|A

A→B|C

B→a

C→D|D e A

D→E B

E→i F t

F→F o G|G

G→b

4. Please show that the following operator grammar is whether an operator precedence grammar by **constructing the related parsing table.** (10%)

   E→E a F|F

   F→F o T|T

   T→(E)|n T|b

5. Please **construct a LR(1) parsing table for the following ambiguous grammar with the additional conditions** that ***,** ⊗ and ⊕ **have the properties of left associative law**, and *** has higher precedence than** ⊗ , ⊗ **has higher precedence than**

⊕.(15%)

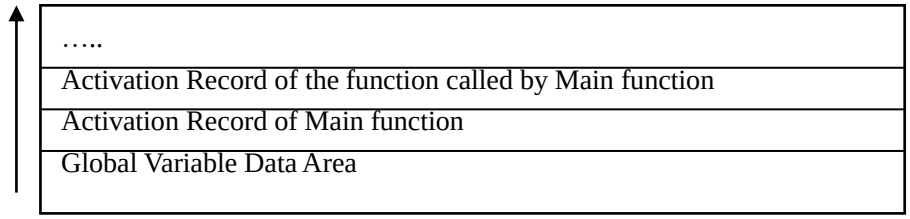
E→E⊕E|E⊗**E|E*|(E)|a|b**

6. Please construct **an annotated parse tree** for the input string 123.123 where the syntax-directed definition is as following (10%):

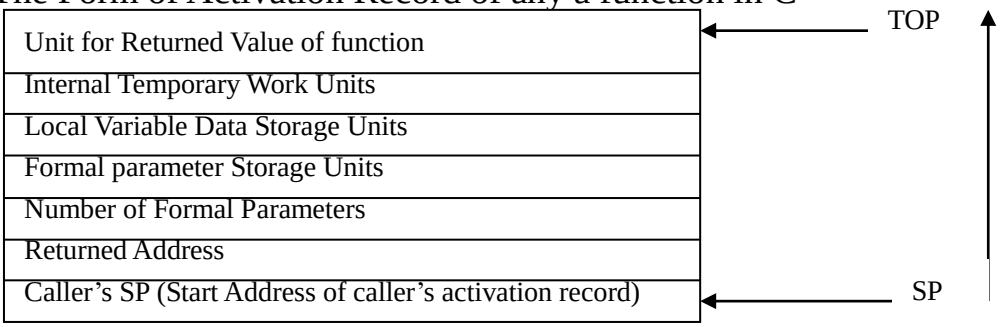
Productions	Semantic Rules
$S \rightarrow L^{(1)}.L^{(2)}$	$S.val=L^{(1)}.val+L^{(2)}.val/4^{L(2).len}$
$S \rightarrow L$	$S.val=L.val$
$L \rightarrow L^{(1)}B$	$L.val=L^{(1)}.val*4+B.val$, $L.len=L^{(1)}.len+1$
$L \rightarrow B$	$L.val=B.val$, $L.len=1$
$B \rightarrow 0$	$B.val=0$
$B \rightarrow 1$	$B.val=1$
$B \rightarrow 2$	$B.val=2$
$B \rightarrow 3$	$B.val=3$

7. We assume that the storage organization and the form of activation record used in C language program run-time stack storage allocation are as following. Please **construct the run-time stack map when it gets the maximum size at the second time** for the following C program (10%).

Storage Organization of C Language



The Form of Activation Record of any a function in C



The C program is as the following:

#include <stdio.h>

int x,y;

```
int main()
{
   x=6;
   y=f(x);
}

int f(int n)
{
   if (n<=1)
         return 1;
   else if(n==2)
          return 2;
        else
         {
          int t1,t2,t3,t;
          t1=f(n-1);
          t2=f(n-2);
```

```
        t3=f(n-3);
        t=t1+t2;
        t=t+t3;
        return t
      }
}
```

Notes: 1) Here we assume that the caller's sp of Main function is the start address of global variable data area, and the returned address in the activation record of a function (including Main function) is filled by the operating system automatically, you might not care it.

2) The initial value of variable X is 6, the start address of stack used in the program is K.

3) The stack map may get its maximum size for several times, here we ask you draw the stack map at maximum size for the second time.

8. Please translate the following program fragment into **three address code sequence, divide the TAC sequence into basic blocks, construct the flow graph and find out all back edges in the flow graph**. (20%)

```
i=1;
while (i<=10) {
   j=1;
   while (j<=10) {
      c[i,j]=0;
      j=j+1
   }
   i=i+1;
}
i=1;
while (i<=10) {
   j=1;
   while (j<=10) {
      k=1;
      while (k<=10) {
```

```
        if (a[i,k]!=0 && b[k,j]!=0)
           c[i,j]=c[i,j]+a[i,k]*b[k,j];
        k=k+1;
    }
    j=j+1;
  }
i=i+1;
}
```

Notes: Here we assume that the declarations of array A,B,C are array [1..10,1..10], each data element of array A,B,C would **use 4 storage unit,** and the start address of array A's storage area is addrA, the start address of array B's storage area is addrB, the start address of array C's storage area is addrC.