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

Surname__________

Student No.__________

Southeast University Examination Paper (A)

Course Name	Principles of Compiling	Examination Term	13-14-2	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%).

   $\{a^m \omega b^m \mid m \geqq 0$ and $\omega \in (c,d,e,f)^*$ and the numbers of d's and e's and f's occurred in ω are **even**$\}$

2. Please construct a **DFA with minimum states** for the following regular expression. (15%)

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

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%)

   S→if E then S| if E then S else S|while E do S|id=F

$E \rightarrow E$ and $E|E$ or $E|$not $E|(E)|b$

$F \rightarrow F+F|F*F|(F)|n$

4. Please **construct a LR(1) parsing table for the following ambiguous grammar with your own defined additional conditions (You determine the required additional conditions by yourself)**.(15%)

$E \rightarrow E\ \theta_1\ E|E\ \theta_2\ E|\ E\ \theta_3\ E|\ (E)|id$

5. Please construct **an annotated parse tree** for the input string (2+3*4+@5)+@6*7 where the syntax-directed definition is as following (10%):

Productions	Semantic Rules
$E \rightarrow E_1 * F$	$\{E.val=E_1.val*F.val\}$
$E \rightarrow F$	$\{E.val=F.val\}$
$F \rightarrow F_1 + G$	$\{F.val=F_1.val+G.val\}$
$F \rightarrow G$	$\{F.val=G.val\}$
$G \rightarrow (E)$	$\{G.val=E.val\}$
$G \rightarrow i$	$\{G.val=i.lexval\}$
$G \rightarrow @i$	$\{G.val=0-i.lexval\}$

6. 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

…..
Activation Record of the function called by Main function
Activation Record of Main function
Global Variable Data Area

The Form of Activation Record of any a function in C

Unit for Returned Value of function	TOP
Internal Temporary Work Units	
Local Variable Data Storage Units	
Formal parameter Storage Units	
Number of Formal Parameters	
Returned Address	
Caller's SP (Start Address of caller's activation record)	SP

```
#include <stdio.h>
int x,y;

 int main()
 {
    x=10;
    y=f(x);
 }

 int f(int m)
{
   if (m>=0)
      if (m==0) return(1)
```

```
        else if (m==1) return(1)
            else if (m==2) return(2)
             else {
                  int t1,t2,t3,t;
                  t1=f(m-1);
                  t2=f(m-2);
                  t3=f(m-3);
                  t=t1+t2+t3;
                  return(t)
             }
      else return(-1)

}
```

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 10, 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.

7. Please translate the following program fragment into **Quadruple sequence using short circuit code and back-patching techniques**. (15%)

```
i=1;
while (i<=10) {
   j=1;
   while (j<=10) {
        k=1;
        m=0;
        while (k<=10) {
             if (A[i,k]==0 || B[k,j]==0) k=k+1;
             else {
                    m=m+A[i,k]*B[k,j];
```

```
            k=k+1
        }
    C[i,j]=m;
    j=j+1;
  }
  i=i+1
  }
```

Notes: 1)Here we assume that the declaration of array A, array B and array C are array [1..10,1..10], each data element of **array A,array B and array 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.

2) The related semantic rules are described as followings, where NXINSTR means "No. of Next Instruction" , the No. of first instruction is (1).

$E \rightarrow i$ {E•TC=NXINSTR; E•FC=NXINSTR+1;

GEN('if' i.place '<>0' 'goto 0'); GEN('goto 0')}

$E \rightarrow E_a$ rop E_b {E•TC=NXINSTR; E•FC=NXINSTR+1;

GEN('if' E_a.place rop.op E_b.place 'goto 0'); GEN('goto 0')}

$E \rightarrow (E^{(1)})$ {E•TC= $E^{(1)}$•TC; E•FC= $E^{(1)}$•FC}

$E \rightarrow$ not $E^{(1)}$ {E•TC= $E^{(1)}$•FC; E•FC= $E^{(1)}$•TC}

$E^A \rightarrow E^{(1)}$ and {BACKPATCH($E^{(1)}$•TC,NXINSTR); E^A•FC= $E^{(1)}$•FC;}

$E \rightarrow E^A E^{(2)}$ {E•TC= $E^{(2)}$•TC; E•FC=MERG(E^A•FC,$E^{(2)}$•FC}

$E^0 \rightarrow E^{(1)}$ or {BACKPATCH($E^{(1)}$•FC,NXINSTR); E^0•TC= $E^{(1)}$•TC;}

$E \rightarrow E^0 E^{(2)}$ {E•FC= $E^{(2)}$•FC; E•TC=MERG(E^0•TC,$E^{(2)}$•TC}

$C \rightarrow$ if E then {BACKPATCH(E•TC,NXINSTR); C•CHAIN=E•FC;}

$T \rightarrow C\ S^{(1)}$ else {q=NXINSTR; GEN('goto 0');

BACKPATCH(C•CHAIN,NXINSTR);

T •CHAIN=MERG($S^{(1)}$•CHAIN,q)}

$S \rightarrow T\ S^{(2)}$ {S•CHAIN=MERG(T•CHAIN,$S^{(2)}$•CHAIN)}

$S \rightarrow C\ S^{(1)}$ {S•CHAIN=MERG(C•CHAIN,$S^{(1)}$•CHAIN)}

$W \rightarrow$ while {W•LABEL=NXINSTR}

$W^d \rightarrow W$ E do {BACKPATCH(E•TC,NXINSTR);W^d•CHAIN=E•FC;

W^d•LABEL=W•LABEL;}

$S \rightarrow W^d\ S^{(1)}$ {BACKPATCH($S^{(1)}$•CHAIN, W^d•LABEL);

GEN('goto' W^d •LABEL); S • CHAIN= W^d•CHAIN}

$A \rightarrow S;$ {BACPATCH(S•CHAIN ,NXINSTR)}

$S \rightarrow AS^{(1)}$ {S•CHAIN = $S^{(1)}$•CHAIN }

(NXINSTR means "No. of Next Instruction")

8. Please **construct the DAG** for the following basic block, optimize the block and **rewrite the block** in optimized code form. Note that we assume **only Variable L would be used later**(10%)

B=3

D=A+C

E=A*C

F=D+E

G=B*F

H=A+C

I=A*C

J=H+I

K=B*5

L=K+J

M=L