Kurdistan Regional Government
Ministry of Higher Education & Scientific Research
University of Salahddin – Hawler
College of Engineering
Software Engineering Dept.

Final Examinations (2011/2012)
Subject: Compilers
Third Year Students
Time allowed: 3 hours
Lecturer: Amanj Sherwany

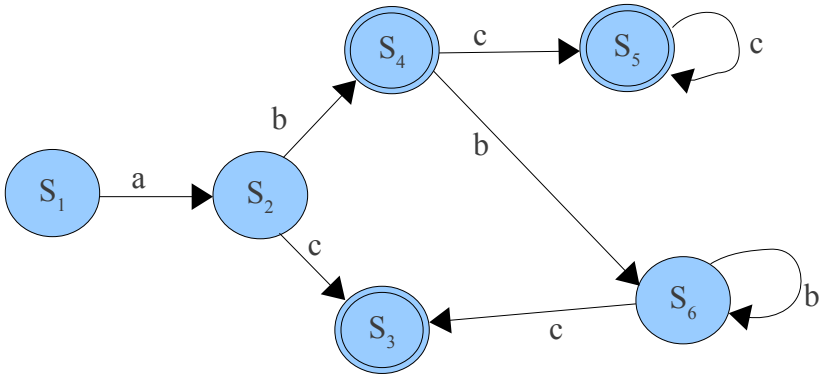
The highest obtainable mark is 100, the minimum passing mark is 50

Q1/ A: (15 points)

Construct a deterministic finite automaton (DFA) for the following regular expression, knowing that the alphabet consists of the symbols *a, b* and *c*.

```
(a(b*)c) | (ab(c*))
```

Answer

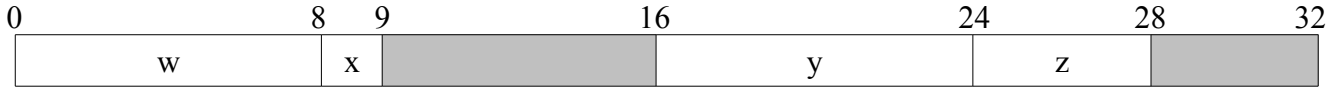


B: (10 points)

```
struct element{double w; char x; double y; int z;}
```

For the above struct, show how its instances are represented in the memory of a 32-bit machine, state its size in bytes and its alignment, then suggest the best way to rearrange the above struct, so that it consumes less memory.

Answer



Size = 32 and alignment = 8

A better way to represent the above struct is:

```
struct element{double w; double y; char x; int z;}
```

* * *

Q2/ A: (10 points)
Consider the following context-free grammar G_0:

$$E \rightarrow E > E \mid E \,\&\, E \mid E = E \mid (E) \mid \mathrm{x}$$

1. Assume that (E & E) has the highest priority, ($E > E$) has the second-highest priority and ($E = E$) has the lowest priority. Rewrite G_0 to an equivalent G_1 which expresses these properties. **(5 points)**
2. Assume that (> and &) are left-associative and (=) is right-associative. Rewrite G_1 to an equivalent grammar G_2 which expresses these properties. **(5 points)**

Answer

1.

$E \rightarrow E = E \mid T$
$T \rightarrow T > T \mid F$
$F \rightarrow F \,\&\, F \mid K$
$K \rightarrow (E) \mid \mathrm{x}$

2.

$E \rightarrow T = E \mid T$
$T \rightarrow T > F \mid F$
$F \rightarrow F \,\&\, K \mid K$
$K \rightarrow (E) \mid \mathrm{x}$

B: (20 points) For the following context-free grammar:

$$S \rightarrow \mathrm{a}A\$ \mid B\$$$
$$A \rightarrow \mathrm{b}B \mid \mathrm{a}$$
$$B \rightarrow \mathrm{c} \mid \mathrm{b}A\mathrm{c}$$

1. Find FIRST and FOLLOW sets for all non-terminals (variables) **(10 points)**
2. Construct LL(1) decision table. **(10 points)**

Answer

1.

S, A and *B* are not nuallable
FIRST(S) = {a, b, c}, FIRST(A) = {a, b}, FIRST(B) = {b, c}
FOLLOW(S) = {}, FOLLOW(A) = {c, $}, FOLLOW(B) = {c, $}

2.

	a	b	c	$
S	$S \rightarrow \mathrm{a}A\$$	$S \rightarrow B\$$	$S \rightarrow B\$$	
A	$A \rightarrow \mathrm{a}$	$A \rightarrow \mathrm{b}B$		
B		$B \rightarrow \mathrm{b}A\mathrm{c}$	$B \rightarrow \mathrm{c}$	

* * *

Q3: (20 points)
Translate the following function to assembly:

```
int fibonacci(int n){
     if (n == 0)
          return 0;
     if (n == 1)
          return 1;
     else
          return (fibonacci(n-1) + fibonacci(n-2));
}
```

The target machine is a 32-bit RISC with 4-byte integers, a stack pointer register SP, a return address register RA, eight general purpose registers R0 to R7, and the following instruction set:

```
     goto label
     if reg < opnd goto label     (==, >=, etc)
     move dst, opnd               assign opnd to dst
     add dst, reg1, opnd2         assign reg1 plus opnd2 to dst
                                  ditto for sub, mul, and, or,
                                  leftshift, etc
     load dst, (reg1 + opnd2)     read integer from memory at
                                  reg1+opnd2
     store (reg1 + opnd1), opnd3  write integer (opnd3) to memory
                                  at reg1+opnd2
     call label                   set RA to next instruction then
                                  jump to label
     return                       jump to the address in RA
```

Each *dst* must be a register, and each *opnd* must be a register or an integer constant. The function call conventions are that parameter is passed in register *R0,* the return value is returned in *R1,* and a function call may destroy any general-purpose register and *RA*. The stack grows from high to low addresses, and *SP* should always point to the lowest word of the current stack frame.
Apart from the statements and expressions, include the code for setting up the stack frame, storing registers in the stack frame , and fetching registers from the stack frame.

Answer

```
        .text
        .global
fibonacci:
# Memory Layout:
# ==============
# 1 * 4: old FP
# 2 * 4: RA
# 3 * 4: n
# 4 * 4: n'

#OUTPUT: R1 and INPUT: R0
        move R7, FP
        move FP, SP
        sub SP, SP, 16

        store FP - 1 * 4, FP
        store FP - 2 * 4, R!
        store FP - 3 * 4, R0

        if R0 != 0 goto LELSEIF
        move R1, 0
        goto Lreturn
LELSEIF:
        if R != 1 goto LELSE
        move R1, 1
        goto Lreturn
LELSE:  sub R0, R0, 1
        call fibonacci
        store FP - 4 * 4, R1
        load R0, FP - 3 * 4
        sub R0, R0, 2
        call fibonacci
        load R2, FP - 4 * 4
        add R1, R1, R2
Lreturn:
        load RA, FP - 2 * 4
        load FP, FP - 1 * 4
        add SP, SP, 16
        return
```

* * *

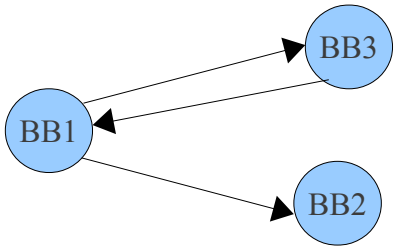
Q 4: (25 points)

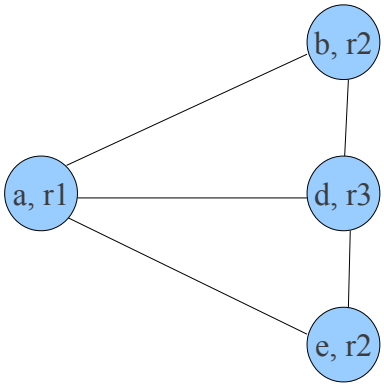
For the following intermediate code representation:

- Find the basic blocks **(3 points)**
- Draw the Control-Flow Graph (CFG) **(2 points)**
- Calculate the Liveness (LIVEIN and LIVEOUT) for each instruction **(15 points)**
- Allocate each of (a, b, d, e) temporary variables in three registers (K = 3). **(5 points)**

Answer

	BBs	**USE**	**DEF**	**LIVEIN**	**LIVEOUT**
I1. L1: a := d + e I2. if a == d goto L2	BB1	d, e a, d	a –	b, d, e a, b, d, e	a, b, d, e a, b, e
I3. d := a – b I4. return d	BB2	a, b d	d –	a, b d	d –
I5. L2: d := a + b I6. goto L1	BB3	a, b –	d –	a, b, e b, d, e	b, d, e b, d, e





Simplify:

push(b) [degree 2]
push(d) [degree 2]
push(e) [degree 1]
push(a) [degree 0]

Select:

select(a), assign a = r1
select(e), assign e = r2
select(d), assign d = r3
select(b), assign b = r2