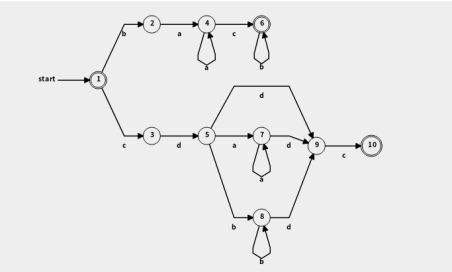
Exercise 1: Scanners

a. Describe informally the language the automaton accepts



b. Do Thompsons construction, subset construction, Hopcroft minimization on the reg ex $(a|b) * cb(a * b * |\varepsilon)$

Exercise 2: Parsers

a. write a context-free grammar for Boolean expressions

b. Use the just contructed grammar. Is it LL(1), if not so make it LL(1).

Exercise 3: IR

```
int[] a = {0,0,0,0,0};
int c = 1;
for (i=0; i < 20; ++i) {
  if (a[4] = 1) {
    c++;
  } else {
    a[i] = c;
  }
}
if (c == 2)
  while (true) {
    c++;
    if (c > 999) {
          break;
        }
  }
} else if (c == 1 ) {
  a[i] = 3;
}
```

a. sketch an @GRAPH

b. for what applications can @GRAPH be used

Exercise 4: Code Shape

```
statement_sequence
|-assignment
|-|-ADDR x: Type as Variable | Offset: 0
|-|-CONSTANT const0: Constant as Variable | Value: 10
|-assignment
|-|-ADDR y: Type as Variable | Offset: 4
|-|-CONSTANT const1: Constant as Variable | Value: 3
|-assignment
|-|-ADDR z: Type as Variable | Offset: 8
|-|-DIV
|-|-|-DEREF x: Type as Variable | Offset: 0
|-|-|-+
|-|-|-|-DEREF y: Type as Variable | Offset: 4
|-|-|-|-CONSTANT const2: Constant as Variable | Value: 1
|-assignment
|-|-ADDR x: Type as Variable | Offset: 0
|-|-*
|-|-|--
|-|-|-|-*
|-|-|-|-|-CONSTANT Pi: Constant as Variable | Value: 3
|-|-|-|-|-DEREF z: Type as Variable | Offset: 8
|-|-|-|-DEREF y: Type as Variable | Offset: 4
|-|-|--
|-|-|-|-*
|-|-|-|-|-CONSTANT Pi: Constant as Variable | Value: 3
|-|-|-|-|-DEREF z: Type as Variable | Offset: 8
|-|-|-|-DEREF y: Type as Variable | Offset: 4
|-assignment
|-|-ADDR y: Type as Variable | Offset: 4
|-|-*
|-|-|-DEREF x: Type as Variable | Offset: 0
|-|-|-DIV
|-|-|-|-CONSTANT const3: Constant as Variable | Value: 1
|-|-|-|-DEREF z: Type as Variable | Offset: 8
```

a. use treewalk code gen assuming unlimited registers

b. Sketch the control flow of the following code snippets

```
int[] a = {0,0,0,0,0};
int c = 1;
for (i=0; i < 20; ++i) {
  if (a[4] = 1) {
    c++;
  } else {
    a[i] = c;
        if (c == 2){
          while (true) {
            c++;
                if (c > 999) {
                  break;
                }
          }
        } else if (c == 1 ) {
          a[i] = 3;
        }
  }
}
if (c == 2)
  while (true) {
    c++;
        if (c > 999) {
          break;
        }
  }
} else if (c == 1 ) {
  a[i] = 3;
}
```

Exercise 5: Instruction Selection & Scheduling

a:	loadAI	r_{arp},@a	⇒ r_1
b:	add	r_1,r_1	⇒ r_1
c:	loadAI	r_{arp},@b	⇒ r_2
d:	mult	r_1,r_2	⇒ r_1
e:	loadAI	r_{arp},@c	⇒ r_3
f:	mult	r_1,r_2	⇒ r_1
g:	loadAI	r_{arp},@d	⇒ r_2
h:	mult	r_1,r_2	⇒ r_1
i:	storeAI	r_1	⇒ r_{arp},@a

a. draw the dependence graph and annotate each node with the cummulative latency

b. use global list scheduling to schedule the code fragment

Exercise 6: Register Allocation

Start	Operations			
1	loadAI	r_{arp}, @a	⇒	r_1
4	add	r_1, r_1	⇒	r_1
5	loadAI	r_{arp}, @b	⇒	r_2
8	mult	r_1, r_2	⇒	r_1
10	loadAI	r_{arp}, @c	⇒	r_2
13	mult	r_1, r_2	⇒	r_1
15	loadAI	r_{arp}, @d	⇒	r_2
18	mult	r_1, r_2	⇒	r_1
20	storeAI	r_1	⇒	r_{arp}, @a

a. write down the live ranges as set of intervals

b. show the result of using the bottom-up global algorithm on it to alloc registers

c. show the result of using the top-down global algorithm on it to alloc registers