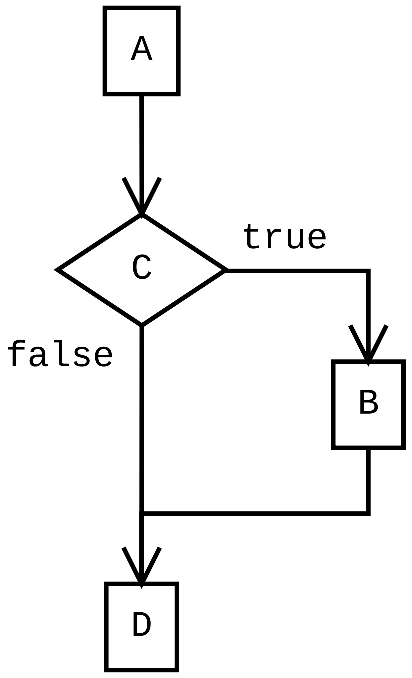
Announcements

Assignment 2 has been posted.

Conditionals

```
A;
if (C) {
    B;
}
D;
```

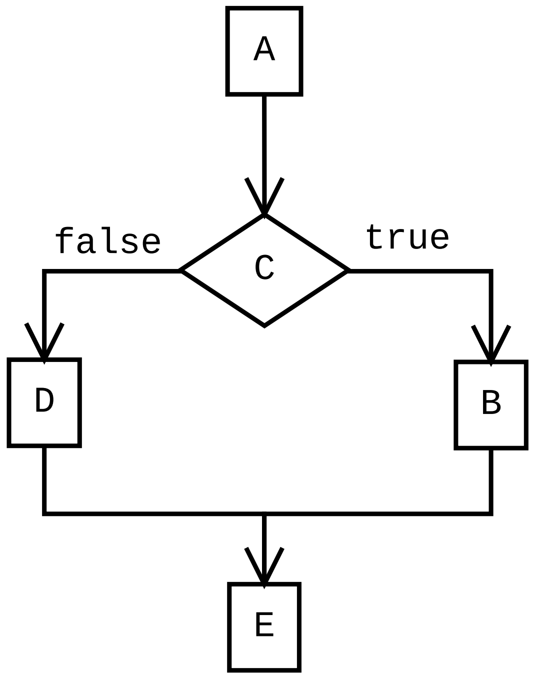
- Control flow diagram



Conditionals

```
A;
if (C) {
    B;
}
else {
    D;
}
E;
```

- Control flow diagram



Some syntactic aspects

```
int n, k = 2;
boolean b = false;
n = Keyboard.readInt();

if (n < 5) {
  b = true;
}
k = 9;
```

is not the same as

```
int n, k = 2;
boolean b = false;
n = Keyboard.readInt();

if (n < 5) {
  b = true;
}
else {
  k = 9;
}
```

Some syntactic aspects

```
int n, k = 2;
boolean b = false;
n = Keyboard.readInt();

if (n < 5)
  b = true;
  k = 9;
```

is the same as

```
int n, k = 2;
boolean b = false;
n = Keyboard.readInt();

if (n < 5) {
  b = true;
}
k = 9;
```

Some syntactic aspects

```
int n, k = 2;
boolean b = false;
n = Keyboard.readInt();

if (n < 5)
  b = true;
  k = 9;
```

is *not* the same as

```
int n, k = 2;
boolean b = false;
n = Keyboard.readInt();

if (n < 5) {
  b = true;
  k = 9;
}
```

Some syntactic aspects

```
int n, k = 2;
boolean b = false;
String s;
n = Keyboard.readInt();
s = Keyboard.readString();
if (n < 5) {
  b = true;
}
else {
  if (s.equals(“one”)) {
    k = 9;
  }
  else {
    k = 7;
  }
}
```

Some syntactic aspects

```
int n, k = 2;
boolean b;
String s;
n = Keyboard.readInt();
s = Keyboard.readString();
if (n < 5) {
b = true;
}
else {
if (s.equals(“one”)) {
k = 9;
}
else {
k = 7;
}
}
```

Some syntactic aspects

```
int n, k = 2;
boolean b;
String s;
n = Keyboard.readInt();
s = Keyboard.readString();
if (n < 5) b = true;
else if (s.equals(“one”))
        k = 9;
else k = 7;
```

Some syntactic aspects

```
int n, k = 2;
boolean b;
String s;
n = Keyboard.readInt();
s = Keyboard.readString();
if (n < 5) b = true;
else k = 9;
else k = 7;  // WRONG!
```

Some syntactic aspects

```
int n, k = 2;
boolean b;
String s;
n = Keyboard.readInt();
s = Keyboard.readString();
if (n < 5)
  if (s.equals(“two”)) b = true;
  else k = 9;
else k = 7;
```

Some syntactic aspects

```
int n, k = 2;
boolean b;
String s;
n = Keyboard.readInt();
s = Keyboard.readString();
if (n < 5)
  if (s.equals(“two”)) b = true;
  else k = 9;
```

Some syntactic aspects

```
int n, k = 2;
boolean b;
String s;
n = Keyboard.readInt();
s = Keyboard.readString();
if (n < 5) {
  if (s.equals(“two”)) b = true;
  else k = 9;
}
```

Some syntactic aspects

```
int n, k = 2;
boolean b;
String s;
n = Keyboard.readInt();
s = Keyboard.readString();
if (n < 5) {
  if (s.equals(“two”)) b = true;
}
else k = 9;
```

Conditionals

- A few properties of conditionals: (`C` is any boolean expression, `P`, `Q`, and `R` are any list of statements.)

  ```
  P;
  if (C) {
      Q;
  }
  R;
  ```

  is equivalent to

  ```
  P;
  R;
  ```

  if the value of `C` is always `false`

Conditionals

- A few properties of conditionals: (`C` is any boolean expression, `P`, `Q`, `R`, and `S` are any list of statements.)

  ```
  P;
  if (C) {
      Q;
  }
  else {
    R;
  }
  S;
  ```

  is equivalent to

  ```
  P;
  R;
  S;
  ```

  if the value of `C` is always `false`

Conditionals

- A few properties of conditionals: (`C` is any boolean expression, `P`, `Q`, and `R` are any list of statements.)

```
P;
if (C) {
    Q;
}
R;
```

is equivalent to

```
P;
Q;
R;
```

if the value of `C` is always `true`

Conditionals

- A few properties of conditionals: (`C` is any boolean expression, `P`, `Q`, `R`, and `S` are any list of statements.)

```
P;
if (C) {
    Q;
}
else {
  R;
}
S;
```

  is equivalent to

```
P;
Q;
S;
```

  if the value of `C` is always `true`

Conditionals

- A few properties of conditionals: (`C` is any boolean expression, `P`, `Q`, and `R` are any list of statements.)

```
P;
if (C == true) {
    Q;
}
R;
```

  is equivalent to

```
P;
if (C) {
    Q;
}
R;
```

Conditionals

- A few properties of conditionals: (`C` is any boolean expression, `P`, `Q`, and `R` are any list of statements.)

```
P;
if (C == false) {
    Q;
}
R;
```

is equivalent to

```
P;
if (!C) {
    Q;
}
R;
```

Properties of conditionals

- In the following, `C` is any boolean expression, `P`, `Q`, `R`, and `S` are any list of statements.

  ```
  P;
  if (C) {
    Q;
  }
  else {
    R;
  }
  S;
  ```

  is equivalent to

  ```
  P;
  if (!C) {
    R;
  }
  else {
    Q;
  }
  S;
  ```

Properties of conditionals

- In the following, C, D are any boolean expressions, P, Q, and R are any list of statements.

```
P;
if (C && D) {
  Q;
}
R;
```

  is equivalent to

```
P;
if (C) {
  if (D) {
    Q;
  }
}
R;
```

Properties of conditionals

- In the following, C, D are any boolean expressions, P, Q, and R are any list of statements.

```
P;
if (C || D) {
  Q;
}
R;
```

is equivalent to

```
P;
if (C) {
  Q;
}
else {
  if (D) {
     Q;
  }
}
R;
```

Properties of conditionals

- Consider the following:

```
int x = 4, y;
String z = "one";
y = Keyboard.readInt();
if (x > 3 && y < 6) {
  y = y + 8;
  z = "two";
}
z = z + "three";
```

is equivalent to

```
int x = 4, y;
String z = "one";
y = Keyboard.readInt();
if (x > 3) {
  if (y < 6) {
    y = y + 8;
    z = "two";
  }
}
z = z + "three";
```

Properties of conditionals

but it is *not* equivalent to

```
int x = 4, y;
String z = “one”;
y = Keyboard.readInt();
if (x > 3) {
  y = y + 8;
  if (y < 6) {
    z = “two”;
  }
}
z = z + “three”;
```

Properties of conditionals

- Consider the following:

```
boolean high = false;
double altitude;
altitude = Keyboard.readDouble();
System.out.println(“Begin”);
if (altitude > 2000.0) {
  high = true;
  System.out.println(“It is high”);
}
else {
  high = true;
  System.out.println(“It is low”);
}
```

Properties of conditionals

- It is equivalent to:

```
boolean high = false;
double altitude;
altitude = Keyboard.readDouble();
System.out.println(“Begin”);
high = true;
if (altitude > 2000.0) {
  System.out.println(“It is high”);
}
else {
  System.out.println(“It is low”);
}
```

Properties of conditionals

- Consider the following:

```
double altitude;
altitude = Keyboard.readDouble();
System.out.println(“Begin”);
if (altitude > 2000.0) {
  altitude = altitude - 500.0;
  System.out.println(“It is high”);
}
else {
  altitude = altitude - 500.0;
  System.out.println(“It is low”);
}
```

Properties of conditionals

- It is *not* equivalent to:

```
double altitude;
altitude = Keyboard.readDouble();
System.out.println(“Begin”);
altitude = altitude - 500.0;
if (altitude > 2000.0) {
  System.out.println(“It is high”);
}
else {
  System.out.println(“It is low”);
}
```

Properties of conditionals

- In the following, C is any boolean expression, P, Q, R , S, and T are any list of statements.

```
P;
if (C) {
  Q;
  R;
}
else{
  Q;
  S;
}
T;
```

Properties of conditionals

is equivalent to

```
P;
Q;
if (C) {
  R;
}
else {
  S;
}
T;
```

if and only if the statements in `Q` do not modify the variables in `C`

Properties of conditionals

- In the following, `C`, `D` are any boolean expressions, `P`, `Q`, `R`, and `S` are any list of statements.

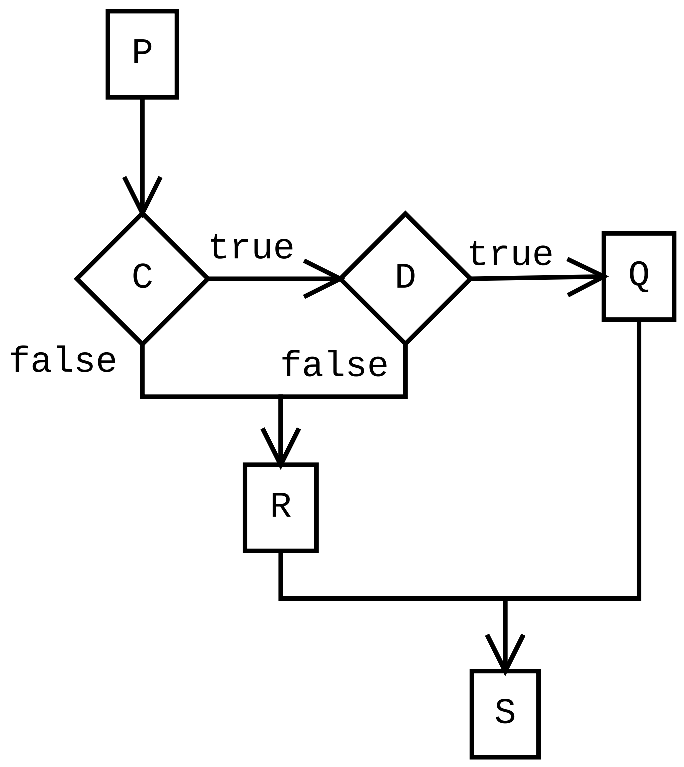
```
P;
if (C && D) {
  Q;
}
else {
  R;
}
S;
```

Properties of conditionals

is equivalent to

```
P;
if (C) {
  if (D) {
    Q;
  }
  else {
    R;
  }
}
else {
  R;
}
S;
```

Properties of conditionals



Properties of conditionals

- In the following, C, D are any boolean expressions, P, Q, R, and S are any list of statements.

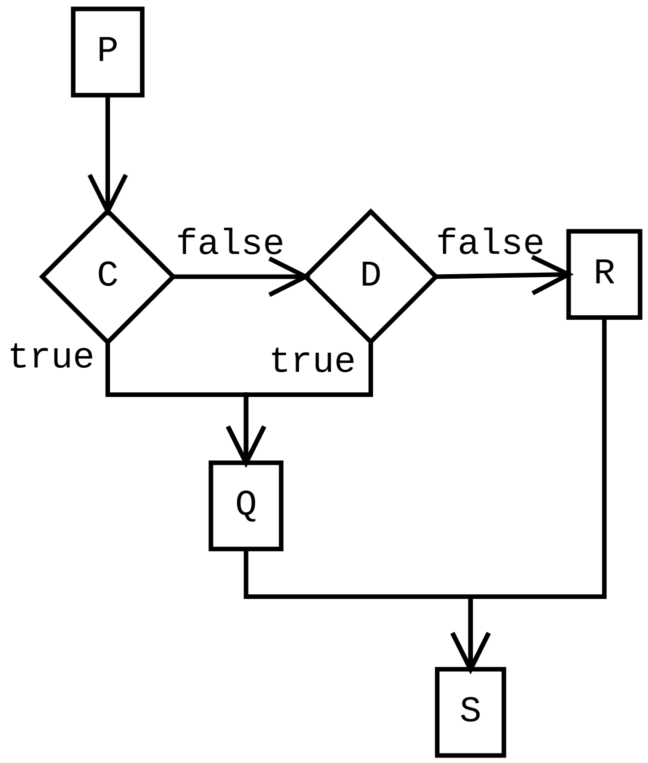
```
P;
if (C || D) {
  Q;
}
else {
  R;
}
S;
```

Properties of conditionals

is equivalent to

```
P;
if (C) {
  Q;
}
else {
  if (D) {
    Q;
  }
  else {
    R;
  }
}
S;
```

Properties of conditionals



Sorting

- Problem: Given three numbers, print them out in ascending order
- Analysis:
  - Input: Three numbers a, b, and c
  - Output: A list of three numbers n_1, n_2, and n_3 taken from a, b, and c, such that it is sorted in ascending order
  - Definitions:
    - A list of three numbers min, mid, and max is sorted in ascending order if the list has the form min, mid, and max, and these numbers satisfy the condition that $min \leq mid$ and $min \leq max$.
  - Requirements: the numbers must be assigned uniquely, that is, the list min, mid, and max must be a *permutation* of the set $\{a, b, c\}$.
  - Assumption: Numbers are comparable

Sorting

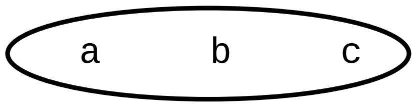
- Design: First alternative: Consider all possibilities

1. If $a \leqslant b$ and $b \leqslant c$ then let min be a, mid be b and max be c
2. If $a \leqslant c$ and $c \leqslant b$ then let min be a, mid be c and max be b
3. If $b \leqslant a$ and $a \leqslant c$ then let min be b, mid be a and max be c
4. If $b \leqslant c$ and $c \leqslant a$ then let min be b, mid be c and max be a
5. If $c \leqslant a$ and $a \leqslant b$ then let min be c, mid be a and max be b
6. If $c \leqslant b$ and $b \leqslant a$ then let min be c, mid be b and max be a

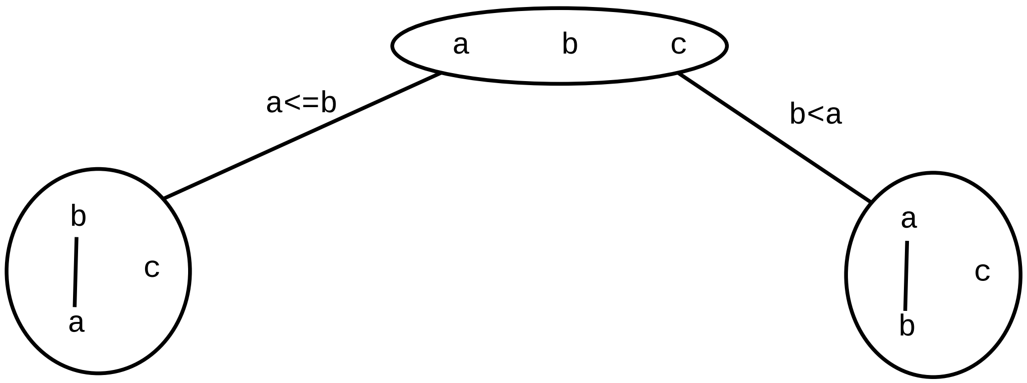
- This solution is correct. It covers all possibilities, but it requires 12 comparisons in the worst case. It is not a very smart solution, and it does not scale well.

Sorting

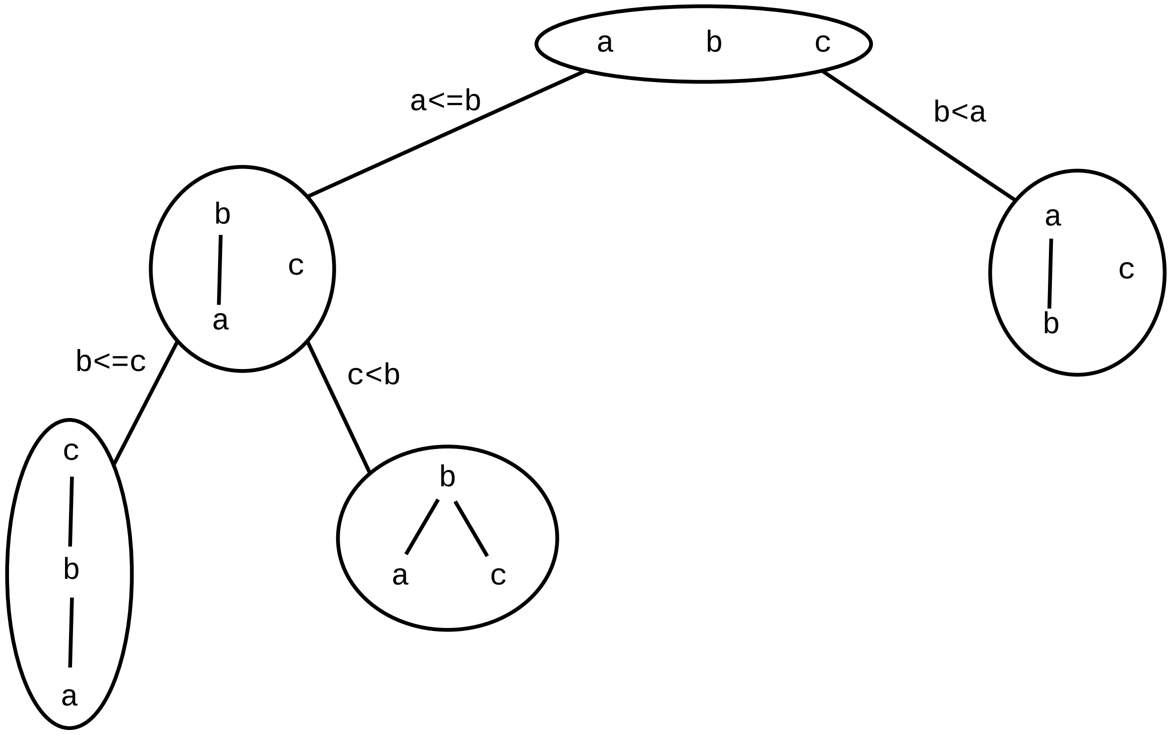
- Second alternative: decision trees



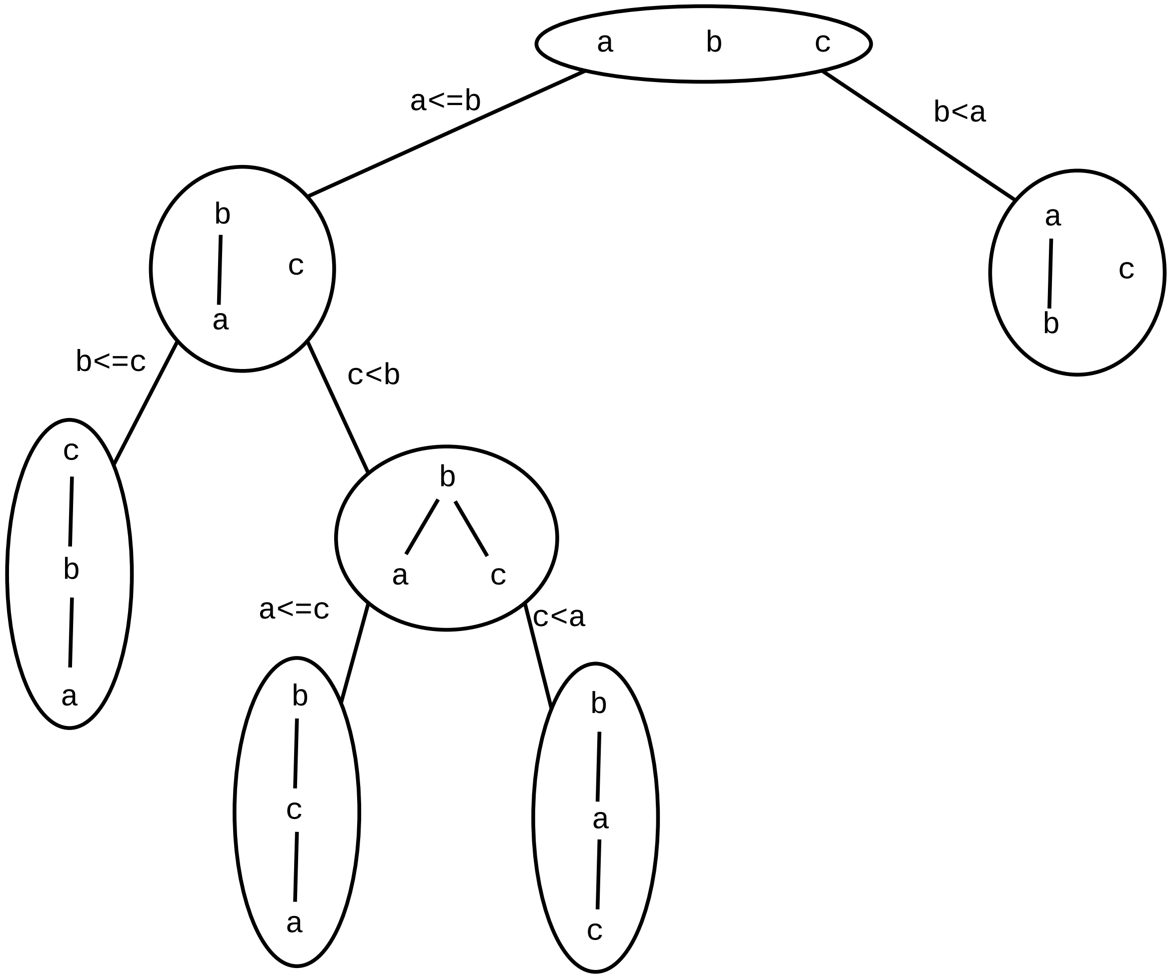
Sorting



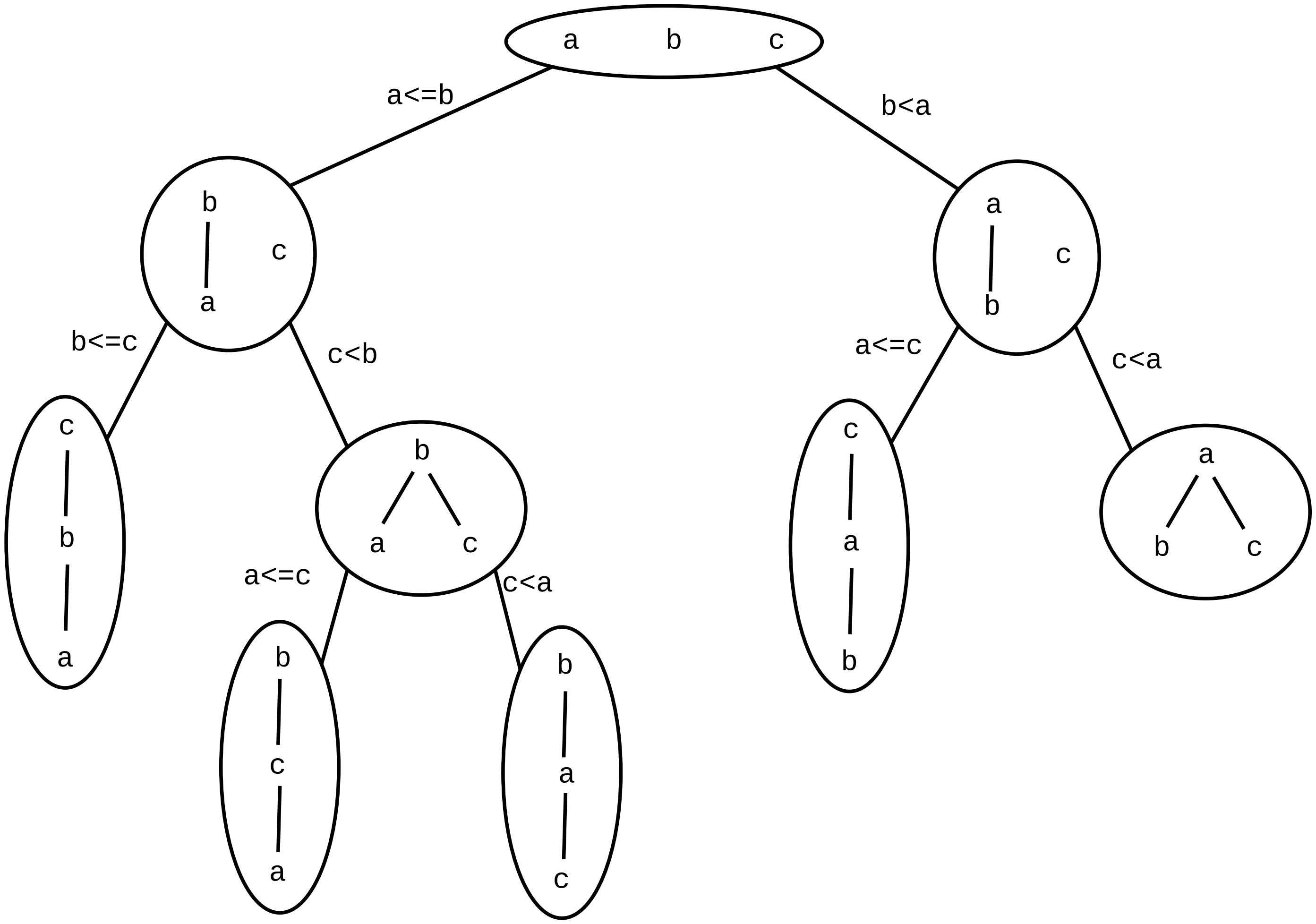
Sorting



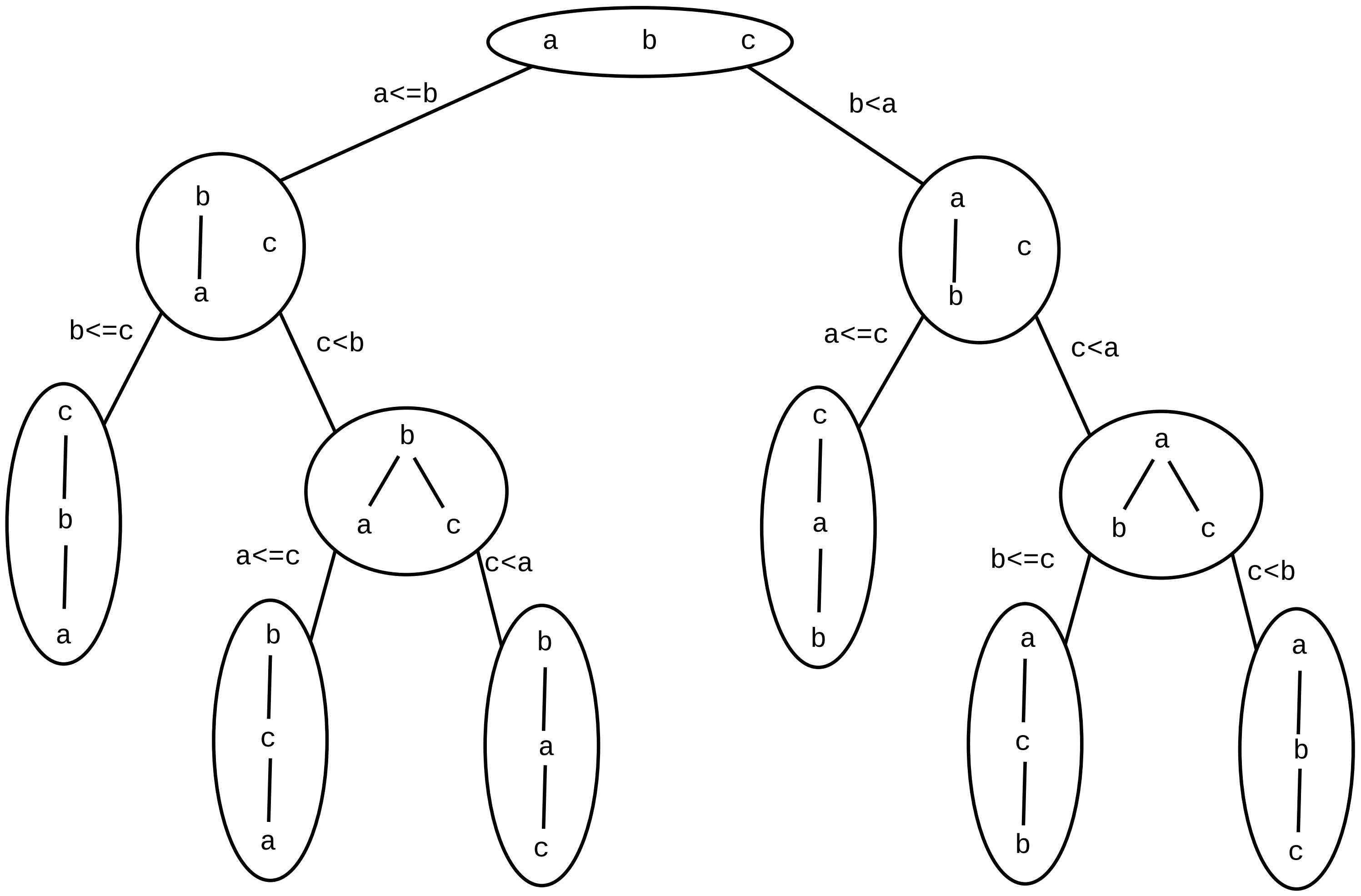
Sorting



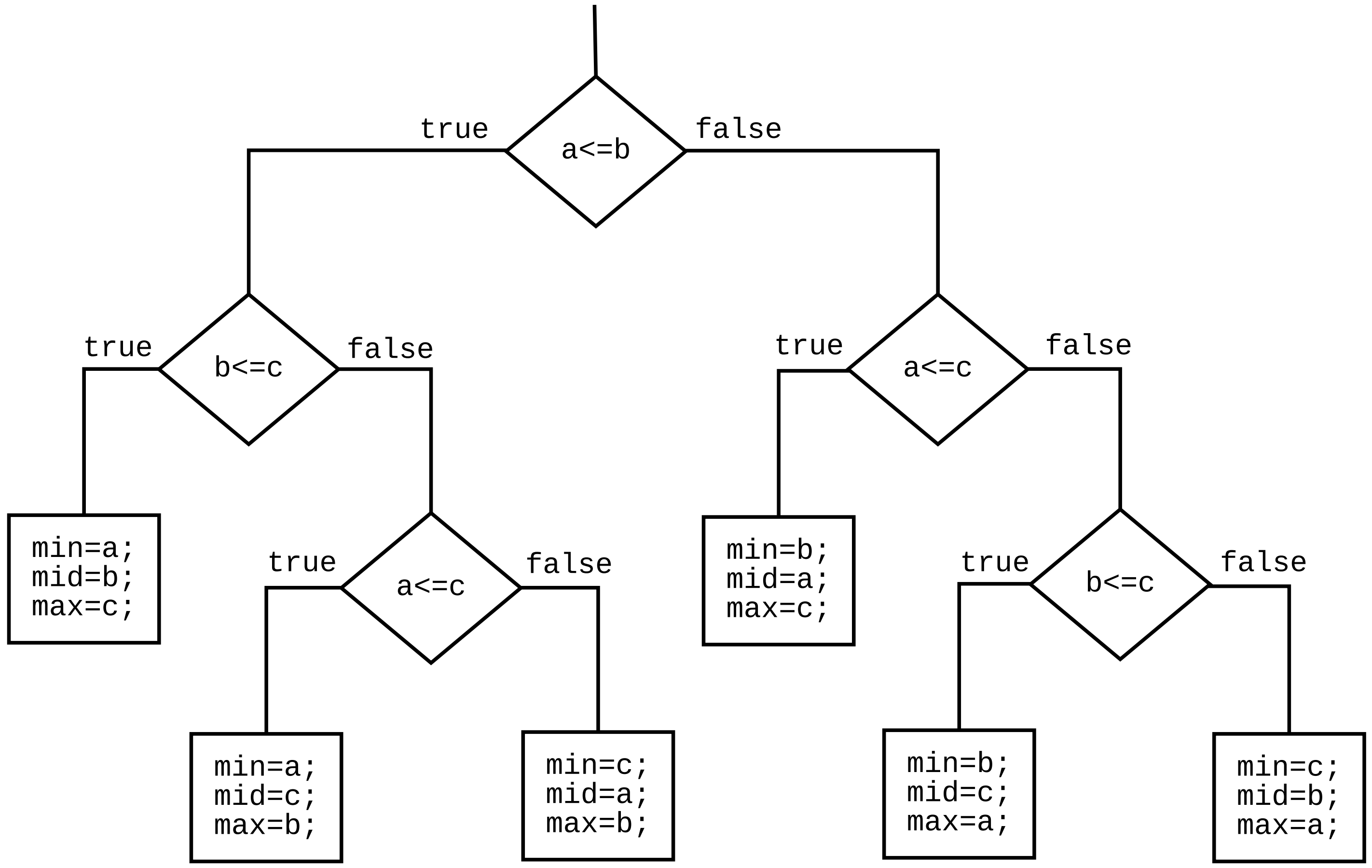
Sorting



Sorting


a b c
a<=b
b<a
b | a c
a | b c
b<=c
c<b
a<=c
c<a
c | b | a
b / a \ c
c | a | b
a / b \ c
a<=c
c<a
b<=c
c<b
b | c | a
b | a | c
a | c | b
a | b | c

Sorting


a<=b
true
false
b<=c
true
false
a<=c
true
false
min=a; mid=b; max=c;
a<=c
true
false
min=b; mid=a; max=c;
b<=c
true
false
min=a; mid=c; max=b;
min=c; mid=a; max=b;
min=b; mid=c; max=a;
min=c; mid=b; max=a;

Sorting

```
import cs1.Keyboard;
public class Sorter {
  public static void main(String[] args)
  {
    double a, b, c, min, mid, max;

    System.out.print(‘‘Enter the first number:’’);
    a = Keyboard.readDouble();
    System.out.print(‘‘Enter the second number:’’);
    b = Keyboard.readDouble();
    System.out.print(‘‘Enter the third number:’’);
    c = Keyboard.readDouble();

    // Continues below ...
```

Sorting

```
if (a <= b) {
  if (b <= c) {
    min = a;
    mid = b;
    max = c;
  }
  else {
    if (a <= c) {
      min = a;
      mid = c;
      max = b;
    }
    else {
      min = c;
      mid = a;
      max = b;
    }
  }
}

// Continues below ...
```

Sorting

```
      else {       // b < a
        if (a <= c) {
          min = b;
          mid = a;
          max = c;
        }
        else {
          if (b <= c) {
            min = b;
            mid = c;
            max = a;
          }
          else {
            min = c;
            mid = b;
            max = a;
          }
        }
      }
      System.out.println(“”’+min+“,’’+mid+“,’’+max);
    } // End of main method
  } // End of Sorter class
```

Sorting

We can make some small changes:

```
if (a <= b) {
  if (b <= c) {
    min = a;
    mid = b;
    max = c;
  }
  else {         // a <= b && c < b
    if (a <= c) {
      min = a;
      mid = c;
      max = b;
    }
    else {
      min = c;
      mid = a;
      max = b;
    }
  }
}
// Continues below ...
```

Sorting

...by "factoring out" the common statement

```
if (a <= b) {
  if (b <= c) {
    min = a;
    mid = b;
    max = c;
  }
  else {          // a <= b && c < b
    if (a <= c) {
      min = a;
      mid = c;
    }
    else {
      min = c;
      mid = a;
    }
    max = b;
  }
}
// Continues below ...
```

Sorting

```
    else {        // b < a
      if (a <= c) {
        min = b;
        mid = a;
        max = c;
      }
      else {
        if (b <= c) {
          min = b;
          mid = c;
          max = a;
        }
        else {
          min = c;
          mid = b;
          max = a;
        }
      }
    }
    System.out.println(“”+min+“,”+mid+“,”+max);
  } // End of main method
} // End of Sorter class
```

Sorting

```
    else {        // b < a
      if (a <= c) {
        min = b;
        mid = a;
        max = c;
      }
      else {                 // b < a && c < a
        if (b <= c) {
          min = b;
          mid = c;
        }
        else {
          min = c;
          mid = b;
        }
        max = a;
      }
    }
    System.out.println(‘“’+min+“‘,’’+mid+“‘,’’+max);
  } // End of main method
} // End of Sorter class
```

Some syntactic shortcuts

- For any variable `v` of a numeric type:

  ```
  v++;
  ```

  is the same as

  ```
  v = v + 1;
  ```

  and

  ```
  v--;
  ```

  is the same as

  ```
  v = v - 1;
  ```

Some syntactic shortcuts

- The ++ and -- operators can be used within expressions (but they shouldn't)

- In this case they can occur in prefex form (++v) or postfix form (v++)

  ```
  x = 2 * v++;
  ```

  is the same as

  ```
  x = 2 * v;
  v = v + 1;
  ```

  and

  ```
  x = 2 * ++v;
  ```

  is the same as

  ```
  v = v + 1;
  x = 2 * v;
  ```

Some syntactic shortcuts

- The `++` and `--` operators can be used within expressions (but they shouldn't)

  ```
  v = 3;
  if (v++ >= 4) System.out.println(``A'');
  ```

  is not the same as

  ```
  v = 3;
  if (++v >= 4) System.out.println(``A'');
  ```

Some syntactic shortcuts

- The ++ and -- operators affect evaluation of conditions

```
v = 4;
if (v++ >= 4 && v < 5) System.out.println(“A”);
```

is not the same as

```
v = 4;
if (v < 5 && v++ >= 4) System.out.println(“A”);
```

The end