4 Decision Making

4.1

(a)
```
if (age<8)
  out.println("Your child must be secured in a car seat.");
```

(b)
```
if (age<8)
  out.println("Your child must be secured in a car seat.");
else
  out.println("Your child may use a regular seat belt.");
```

4.2

```
if (monthly_sales > 500) {
  rate = 7;
  pay = monthly_sales * rate / 100;
} else {
  rate = 5;
  pay = monthly_sales * rate / 100;
}
```

Notice that the computation for `pay` is the same for either case.

4.3

```
if (term <= 24)
  rate = 0.9;
else if (term == 36)
  rate = 2.9;
else if (term == 48)
  rate = 4.9;
else
  rate = 6.9;
```

4.4

```
if (monthly_sales > 1000)
  rate = 10;
else if (monthly_sales > 500)
  rate = 7;
else
  rate = 5;
pay = monthly_sales * rate / 100;
```

4.5

```
if (score >= 90)
  out.println("A");
else if (score >= 80)
  out.println("B");
else if (score >= 60)
  out.println("C");
else if (score >= 50)
  out.println("D");
else
  out.println("F");
```

4.6

```
if (ageOfEntrant >= 65)
  out.println("$1.50");
else if (ageOfEntrant >= 5)
  out.println("$2.50");
else
  out.println("Free");
```

4.7

```
if (X < Y)
  if (Y < Z)
    Max = Z;
  else
    Max = Y;
```

```
else
  if (X < Z)
    Max = Z;
  else
    Max = X;
```

The code will still compute `Max` properly, even if two of `X`, `Y`, and `Z` are the same.

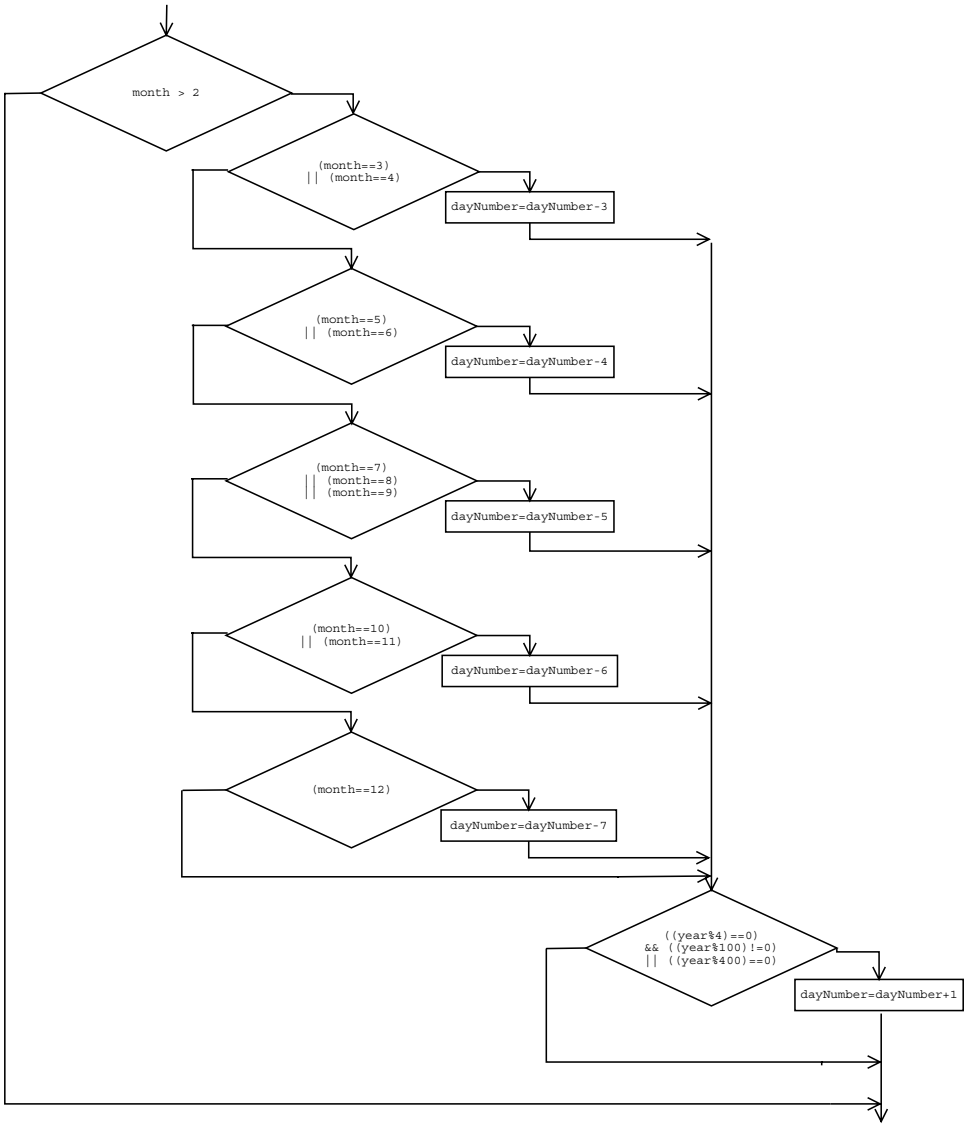
4.8

```
(age>=25) && (age<=40)
          && !smoking && (height<66) && (weight<140)
          && goodLooking && ableToRelocate
```

4.9

```
result = (x <= 1.0) || ((x-1.0) <= 1.0e-5);
```

4.10



4.11

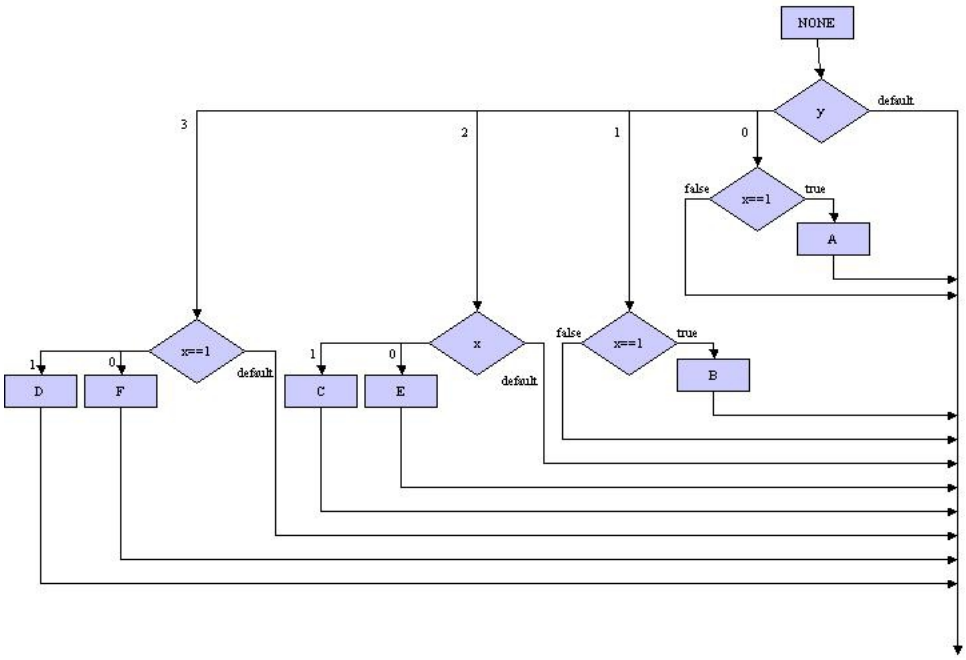
`"wish".compareTo("wishbone");` returns a negative integer.
`"wish you were here".compareTo("wishbone");` returns a negative integer.
`"wash".compareTo("Washington");` returns a positive integer.
`"wash".compareToIgnoreCase("Washington");` returns a negative integer.
`"wash and wear".compareTo("wash-and-wear");` returns a negative integer.

4.12

```
switch (x) {
  case 0:  s = 0;
  default: if (x<0)
             s = -1;
           else
             s = 1;
}
```

The above code sets `s` to `-1` if `x` is negative, and to `1` if `x` is zero or positive. There should be a `break` statement at the end of case `0`.

4.13



4.14

The *nand* operation is `!(p&&q)`

4.15

Proof that `!(p && q) == (!p) || (!q)`.

Input Values		Expression 1		Expression 2		
`p`	`q`	`p && q`	`!(p && q)`	`!p`	`!q`	`(!p) || (!q)`
True	True	True	False	False	False	False
True	False	False	True	False	True	True
False	True	False	True	True	False	True
False	False	False	True	True	True	True