1. There are four violations:

   1. The `then` keyword is not used in Scala (static error).
   2. Local variable `s` must be initialized as part of its definition (static error).
   3. The first branch of the `if` evaluates to `Unit`, which does not match `Int`, the type of the second branch and the declared return type of the function (static error).
   4. Since `j = -1`, the reference `a(j)` will be out of bounds (checked runtime error).

Note that Scala has no unchecked runtime errors.

2. (a) `int`  
   (b) `(a₁ → (a₂ → a₁))`  
   (c) `((bool → a₁) → a₁)`  
   (d) Not typable: `x` cannot be both a function and an integer.  
   (e) `((a₁ → bool) → (a₁ → bool))`

3. Since either arm of the `catch` might yield the overall value, the arms must have the same type, but are otherwise unconstrained (just as for an `if` expression). Since `throw` never actually yields a value, it can safely be assigned any type whatever, and we need that flexibility in order to use it in arbitrary positions in the code.

\[ \text{TE} \vdash (\text{throw}) : t \]

\[ \text{TE} \vdash e₁ : t \quad \text{TE} \vdash e₂ : t \]

\[ \text{TE} \vdash (\text{catch} \ e₁\ e₂) : t \]

4. (a)  
   OO programmer hacks classes  
   Functional programmer uses pattern matching

(b)  
   Scala programmer hacks code  
   Scala programmer hacks code

5.  

```scala
case class P(i:Int, u:T, v:T) extends T {  
    def f() = i * u.f() + v.f()
}

case class Q(b:Boolean) extends T {  
    def f() = if (b) 1 else 0
}
```
6.a. f: x, z. g: y.

(b)

```scala
def M2(x:Boolean, y:Int, z:Int) =
  R(w => if (x) z + w else w - 42,
    w => w + y)
```

7. Under method A, \(s \text{ ++ } "x"\) and \("x" \text{ ++ } s\) will both take time proportional to \(|s|\), because the entire string must be copied. Under method B, \(s \text{ ++ } "x"\) will still take time proportional to \(|s|\), because the string must be traversed, but \("x" \text{ ++ } s\) will take only unit time. So comparing the execution times of the following programs should do the trick: if program 1 runs much faster than program 2, method B is being used; if the runtimes are about the same, it's method A.

Program 1:

```scala
s = ""
for i = 1 to 1000000 do
  s = "x" ++ s;
```

Program 2:

```scala
s = ""
for i = 1 to 1000000 do
  s = s ++ "x";
```

8.

```scala
def count(b:B, x:A) : Int = b match {
  case EmptyB => 0
  case InsertB(b, y) => count(b, x) + (if (x == y) 1 else 0)
  case DeleteB(b, y) => (count(b, x) - (if (x == y) 1 else 0)) max 0
  case UnionB(b1, b2) => count(b1, x) + count(b2, x)
}
```