5.1: Algol 60 Procedure Types

اگر خود پروسه Q دچار مشکل باشد، به run-time error برمی‌خوریم؛ مثال:

```
proc q(b):
    begin
        boolean b;
        ans := b+1;
    end;
```

چون Q یک پروسه است و به عنوان پارامتر به P داده می‌شود، بدنه‌اش بررسی نمی‌شود؛ اما هنگام run-time در ارزیابی عبارت ans := b+1 به یک type error برمی‌خوریم.

5.2: Algol 60 Pass-By-Name

طبق Algol 60 copy rule داریم:

```
i = 1, x = A[i] = A[1] = 2
i = x          #i = A[1] = 2
i = 2          #x = A[2] = 3
x := i         #A[2] = 2
print(i, A[1], A[2]) #2, 2, 2
```

5.3: Nonlinear Pattern Matching

(a)

```
fun f(x) = if y = 0 then x else if x = 0 then y else x+y
```

(b)

خیر؛ همان‌طور که در صورت سوال آمده‌است، الگوهای ML نمی‌توانند شامل متغیر تکراری ((eq(x, x)) باشند.

(c)

کافی است به جای استفاده دوباره از نام یک متغیر، از نام‌های دیگر برای همان متغیر در پارامترهای بعدی استفاده کنیم و سپس برابری آن‌ها را بررسی کنیم؛ بدیهی است که این برابری می‌تواند برابری مقدار یا آدرس حافظه و یا هر دو باشد.

(d)

از آن‌جایی که نمی‌توان برابری توابع را در ML بررسی کرد اما توابع می‌توانند به عنوان پارامتر مورد استفاده قرار بگیرند، طراحان ML با جلوگیری از امکان تکرار متغیرها در الگوها، سعی می‌کنند تا از این موضوع پیش‌گیری کنند.

5.6: Currying

(a)

```
Val Curry = fn f => fn x => f(x, y);
Val UnCurry = fn f => fn (x, y) => (f(x))(y);
```

(b)

```
Curry(f) = fn f => fn x => fn y => f(x, y);
UnCurry(Curry(f)) = fn (x, y) => Curry(f)(x)(y) =>* f(x, y);

UnCurry(f) = fn (x, y) => ((f(x)))(y);
Curry(UnCurry(f)) = fn x => fn y => UnCurry(f)(x, y) =>* f(x, y);
```

پس دو تابع نوشته‌شده مطابق خواسته سوال عمل می‌کنند؛ یعنی دو تایپی که در صورت سوال آمده است را به یکدیگر تبدیل می‌کنند.

$f = ‘a * ‘b \rightarrow ‘c$
$UnCurry(Curry(‘a * ‘b \rightarrow ‘c)) = UnCurry(‘a \rightarrow ‘b \rightarrow ‘c) = ‘a * ‘b \rightarrow ‘c = f$

$g = ‘a \rightarrow ‘b \rightarrow ‘c$
$Curry(UnCurry(‘a \rightarrow ‘b \rightarrow ‘c)) = UnCurry(‘a * ‘b \rightarrow ‘c) = ‘a \rightarrow ‘b \rightarrow ‘c = g$

بنابراین درستی آن اثبات می‌شود.

5.7: Disjoint Unions

(a)

از آن جایی که در یک union تمام متغیرها در یک فضا ذخیره می‌شوند، اگر با یک نوع کار کرده باشیم و بخواهیم با نوع دیگر کار کنیم، ممکن است دچار مشکل run-time شویم.
مثلاً اگر در مثال سوال ابتدا s را که یک char * است برابر "here, fido" قرار دهیم و بعد x.i را به‌علاوه ۵ کنیم، به type-check error برمی‌خوریم.

(b)

در زبان ML هم ممکن است این مشکلات مانند زبان C پیش بیایند.
با استفاده از tagها می‌توان استفاده نادرست را به اطلاع برنامه‌نویس رساند؛ در صورت استفاده اشتباه از یک تایپ، چون نوع آن از قبل مشخص شده‌است، می‌توان به برنامه‌نویس اخطار داد.