

Solutions to the 2002 Sample Exam for CSE3322

Question 1

- (a) 3
- (b) 3
- (c) 5, the value is ~1
- (d) 3
- (e) 5, used to hide the definition of a datatype
- (f) 2
- (g) 1
- (h) 5, it will return $Y = []$
- (i) 3
- (j) 5, 22 will be written
- (k) 3
- (l) 3
- (m) 5
- (n) 4
- (o) 4.

Question 2

```
fun digitToString i = str (chr (i + ord("#0")));  
( * or  
fun digitToString 0 = "0"  
  | digitToString 1 = "1"  
  | digitToString 2 = "2"  
  | digitToString 3 = "3"  
  | digitToString 4 = "4"  
  | digitToString 5 = "5"  
  | digitToString 6 = "6"  
  | digitToString 7 = "7"  
  | digitToString 8 = "8"  
  | digitToString 9 = "9" ;* )
```

```

fun posIntToString i =
  if i < 10 then digitToString i
  else (posIntToString (i div 10))^
       (digitToString (i mod 10));

fun intToString i =
  if i < 0 then "~"^posIntToString (~i)
  else posIntToString i;

```

Question 3

```

datatype FS = File of string * (char list)
            | Directory of string * (FS list)

fun name (File(n,_)) = n^" "
  | name (Directory(n,_)) = n^" "

fun ls (Directory(_,cs)) = concat (map name cs)
  | ls f = name f

```

Question 4

- (a) Call-by-name parameter passing works by textually replacing the formal parameters in the abstraction body by the actual parameters.
- (b) Algol 60 and C macros both use call by name parameter passing.
- (c) It isn't used because it is hard to understand behavior of imperative programs (i.e. programs which update variable values.) As an example consider the program from the lecture notes

```

void swap(int x, y)
{
  int t;
  t := x; x := y; y := t;
}

```

which has strange behaviour with the call `swap(i,a[i])`.

Question 5

(a)

$$\begin{aligned}
 FIRST(S) &= FIRST(X) + \{d, \epsilon\} = \{a, c, d, e, \epsilon\} \\
 FIRST(X) &= FIRST(Y) + FIRST(Z) + \{a\} = \{a, c, e\} \\
 FIRST(Y) &= \{c\} \\
 FIRST(Z) &= \{e\} \\
 FOLLOW(S) &= \{\$\} \\
 FOLLOW(X) &= FIRST(S) - \{\epsilon\} + FOLLOW(S) = \{a, c, d, e, \$\} \\
 FOLLOW(Y) &= FOLLOW(X) = \{a, c, d, e, \$\} \\
 FOLLOW(Z) &= \{b\} + FOLLOW(Y) = \{a, b, c, d, e, \$\}
 \end{aligned}$$

(b) The parsing table is:

	a	b	c	d	e	\$
S	P1		P1	P2	P1	P3
X	P6		P4		P5	
Y			P7			
Z					P8	

the productions are numbered in their original order:

- (1) $S \rightarrow X S$
- (2) $S \rightarrow d S$
- (3) $S \rightarrow \epsilon$
- (4) $X \rightarrow Y$
- (5) $X \rightarrow Z b$
- (6) $X \rightarrow a Y$
- (7) $Y \rightarrow c Z$
- (8) $Z \rightarrow e$

(c) Yes, the table shows no conflicts.

(d) The parsing for *dace* proceeds as follows:

$S\$$	$dace\$$	$P2$
$dS\$$	$dace\$$	adv
$S\$$	$ace\$$	$P1$
$XS\$$	$ace\$$	$P6$
$aYS\$$	$ace\$$	adv
$YS\$$	$ce\$$	$P7$
$cZS\$$	$ce\$$	adv
$ZS\$$	$e\$$	$P8$
$eS\$$	$e\$$	adv
$S\$$	$\$$	$P3$
$\$$	$\$$	$accept$

Question 6

(a) The parser would not be able to decide which of the productions $X \rightarrow Y$ or $X \rightarrow Z b$ to use when trying to expand an X .

(b) The productions from (a) need to be modified such that they use distinct lookahead. This is done by “unfolding” the productions for Y and Z into those for X . The modified grammar is:

$$\begin{aligned}
 S &\rightarrow XS \mid dS \mid \epsilon \\
 X &\rightarrow cZ \mid eb \mid acZ \\
 Z &\rightarrow e
 \end{aligned}$$

Note that you could also expand the Z production into X completely.

$$\begin{aligned}
 S &\rightarrow XS \mid dS \mid \epsilon \\
 X &\rightarrow ce \mid eb \mid ace
 \end{aligned}$$

Question 7

(a)

The collection of sets of LR(0) items is constructed as follows:

$$\begin{aligned}
 I_0 &= \{ \\
 &\quad S' \rightarrow \cdot S \\
 &\quad S \rightarrow \cdot a X \\
 &\quad \} \\
 goto(I_0, S) &= I_1 \\
 &= \{ \\
 &\quad S' \rightarrow S \cdot \\
 &\quad \} \\
 goto(I_0, a) &= I_2 \\
 &= \{ \\
 &\quad S \rightarrow a \cdot X, \\
 &\quad X \rightarrow \cdot b X, \\
 &\quad X \rightarrow \cdot b Y, \\
 &\quad \} \\
 goto(I_2, X) &= I_3 \\
 &= \{ \\
 &\quad S \rightarrow a X \cdot \\
 &\quad \} \\
 goto(I_2, b) &= I_4 \\
 &= \{ \\
 &\quad X \rightarrow b \cdot X \\
 &\quad X \rightarrow b \cdot Y \\
 &\quad X \rightarrow \cdot b X, \\
 &\quad X \rightarrow \cdot b Y, \\
 &\quad Y \rightarrow \cdot c, \\
 &\quad \} \\
 goto(I_4, X) &= I_5 \\
 &= \{ \\
 &\quad X \rightarrow b X \cdot \\
 &\quad \} \\
 goto(I_4, Y) &= I_6 \\
 &= \{ \\
 &\quad X \rightarrow b Y \cdot \\
 &\quad \} \\
 goto(I_4, c) &= I_7 \\
 &= \{ \\
 &\quad Y \rightarrow c \cdot \\
 &\quad \}
 \end{aligned}$$

(b) $FOLLOW(S) = FOLLOW(X) = FOLLOW(Y) = \{\$\}$.

The SLR table is

state	action				goto		
	a	b	c	\$	S	X	Y
0	s2				1		
1				accept			
2		s4				3	
3				r1			
4		s4	s7			5	6
5				r2			
6				r3			
7				r4			

Productions are again numbered in their order in the augmented grammar:

- (0) $S' \rightarrow S$
- (1) $S \rightarrow a X$
- (2) $X \rightarrow b X$
- (3) $X \rightarrow b Y$
- (4) $Y \rightarrow c$

(c)

STACK	INPUT	ACTION
0	<i>a b b c</i> \$	shift 2
0 <i>a</i> 2	<i>b b c</i> \$	shift 4
0 <i>a</i> 2 <i>b</i> 4	<i>b c</i> \$	shift 4
0 <i>a</i> 2 <i>b</i> 4 <i>b</i> 4	<i>c</i> \$	shift 7
0 <i>a</i> 2 <i>b</i> 4 <i>b</i> 4 <i>c</i> 7	\$	reduce 4, goto 6
0 <i>a</i> 2 <i>b</i> 4 <i>b</i> 4 <i>Y</i> 6	\$	reduce 3, goto 5
0 <i>a</i> 2 <i>b</i> 4 <i>X</i> 5	\$	reduce 2, goto 3
0 <i>a</i> 2 <i>X</i> 3	\$	reduce 1, goto 1
0 <i>S</i> 1	\$	accept