CSC488S/CSC2107S Compilers and Interpreters **Winter 2013/2014**

Mid Term Test Solution

1. [20 marks]

a) Could be handled lexically or syntactically but syntactically is much easier.
Note the implicit (true) assumption that two identifiers can never be adjacent (i.e. without an operator or reserved word in between). This is necessary for white space containing identifiers to be unambiguous

b) **Syntactic solution**
Lexical analysis returns ordinary white space delimited identifiers.
Define the syntactic rule:

```
identifier : simpleIdentifier ,
             identifier simpleIdentifier ;
```

and let the parser handle the concatenation.

Lexical solution
Create an identifier buffer in the lexical analyzer.
When the lexical analyzer first encounters an identifier it goes in the buffer.
When the lexical analyzer encounters subsequent identifiers they get concatenated onto the buffer.
When the lexical analyzer encounters a non-identifier, and the identifier buffer is not empty, output the identifier buffer as one identifier token and reset the identifier buffer to empty.

2. [20 marks] Full solution is too complicated to reproduce here. Any reasonably complete attempt to describe the AST received good marks.

3. [20 marks]

a) Replacing **result** with **return** creates an ambiguity, can't distinguish **return** *expression* from **return** followed by an assignment statement.
b) Removing **func** creates ambiguity between parameter lists and argument lists.
c) Harmless (and useless) change.
d) Removing parentheses in conditional expression creates many conflicts, similar to dangling **else** problem

4. [20 marks] Common problems

- Not describing the layout of the two minor structures
- Not allocating space for the control block at the start of the activation record
- Not allocating space for the parameters start, end and X
- Not putting padding between the elements of the arrays V3 and limits so that each array element was properly aligned.

The table below shows the complete layout. fill(N) means N bytes of fill.

Bit Offset	Byte Offset	4 bytes	4 bytes	Bits mod 64	Bytes mod 8
0	0	control	control		
64	8	control	control		
128	16	start, fill(2)	end		
192	24	X	X		
256	32	working[0 , 1]	working[2 , 3]		
320	40	working[4 , 5]	working[6 , 7]		
384	48	working[8 , 9]	i , j		
448	56	k , fill(2)	fill(4)		
512	64	V1	V1	0	0
576	72	V1	V1		
640	80	V1 (2) , fill(2)	fill(4)		
704	88	V2	V2	0	0
768	96	V2	V2		
832	104	V2 (2) , fill(2)	fill(4)		
896	112	V3[0]	v3[0]	0	0
960	120	V3[0]	v3[0]		
1024	128	V3[0] (2) , fill(2)	fill(4)		
1088	136	V3[1]	v3[1]	0	0
1152	144	V3[1]	v3[1]		
1216	152	V3[1] (2) , fill(2)	fill(4)		
1280	160	V3[2]	v3[2]	0	0
1344	168	V3[2]	v3[2]		
1408	176	V3[2] (2) , fill(2)	fill(4)		
1472	184	maxVBlue	MaxVBlue	0	0
1536	192	limit[0]	limit[0]	0	0
1600	200	limit[0]	limit[0]		
1664	208	limit[0]	limit[0]		
1728	216	limit[0]	fill(4)		
1792	224	limit[1]	limit[1]	0	0
1856	232	limit[1]	limit[1]		
1920	240	limit[1]	limit[1]		
1984	248	limit[1]	fill(4)		
2048	256	limit[2]	limit[2]	0	0
2112	264	limit[2]	limit[2]		
2176	272	limit[2]	limit[2]		
2240	280	limit[2]	fill(4)		
2304	288	limit[3]	limit[3]	0	0
2368	296	limit[3]	limit[3]		
2432	304	limit[3]	limit[3]		
2496	312	limit[3]	fill(4)		
2560	320			0	0

5. [20 marks] There were many possible ways to present these tables.
Common problems: incompleteness, missing links for structures.