PHASE 1

Compiler Course Fall-2016

Saman Ferki
9231075

Parham Alvani
9231058

Lex Regular Expressions

```
t_RECORD_KW = r'record'
t_IF_KW = r'if'
t_THEN_KW = r'then'
t_ELSE_KW = r'else'
t_SWITCH_KW = r'switch'
t_CASE_KW = r'case'
t_END_KW = r'end'
t_WHILE_KW = r'while'
t_DEFAULT_KW = r'default'
t_RETURN_KW = r'return'
t_BREAK_KW = r'break'
t_STATIC_KW = r'static'
t_NOT_KW = r'not'
t_AND_KW = r'and'
t_OR_KW = r'or'

t_TRUE = r'true'
t_FALSE = r'false'

t_SEMICOLON = r';'
t_COLON = r':'
t_DOT = r'\.'
t_COMMA = r','

t_BR_OPEN = r'\{'
t_BR_CLOSE = r'\}'
t_PR_OPEN = r'\('
t_PR_CLOSE = r'\)'
t_BK_OPEN = r'\['
t_BK_CLOSE = r'\]'

t_COMMENTS = r'\/\/.*'


t_INT_T = r'int'
t_BOOL_T = r'bool'
t_REAL_T = r'real'
t_CHAR_T = r'char'

t_REL_OP = r'\.eq | \.gt | \.ge | \.lt | \.le | \.ne'
t_MATH_OP = r'\+ | \- | \* | \/ | % | \?'
t_EXP_OP = r'='

t_ID = r'\#[a-zA-Z]{2}[0-9]{2}'
t_FAKE_ID = r'\#[a-zA-Z]{2}[0-9]{2}[\w]+'

t_CHARCONST = r"'\\?[\w'\\]'"

t_REALCONST = r'\d*\.\d+'

t_NUMCONST = r'\d+'

t_ignore = ' \t\r\f\v'
```

Sample Source Code

```
record #po11 {
        int #xx11, #yy11;
}

record #li11 {
        #po11 #xx11, #yy11;
}

int #at11 (int #ba12, #ca23[]; bool #do43, #el32; int #fo12)
{
        int #gn11, #ho12[100];
        real #el72;
        #el72 = 72.20;
        #po11 #aP11; #li11 #aL11;
        #li11 #tw33[2];
        #aP11.#xx11 = 666; #aP11.#yy11 = 667;
        #aL11.#xx11.#xx11 = 1; #aL11.#xx11.#yy11 = 2; #aL11.#yy11.#xx11 = 3; #aL11.#yy11.#yy11 = 4;
        #tw33[0].#xx11.#xx11 = 42; #tw33[1].#yy11.#xx11 = 43;
        #gn11 = #ho12[2] = 3** #ca23; // hog is 3 times the size of array passed to cat
        if (#do43 and #el32 or #ba12 .gt #ca23[3]) #do43 = not #do43;
        else #fo12++;
        if (#ba12 .le #fo12) {
        while (#do43) {
                static int #ho12; // hog in new scope
                #ho12 = #fo12;
                #do43 = #fr77(#fo12++, #ca23) .lt 666;
                if (#ho12 .gt #ba12) break;
        else if (#fo12 .ne 0) #fo12 += 7;
```

```
        }
        }
        #fo12 = ?5;
        switch (#fo12) {
                case 0:
                        #fo12++;
                        break;
                case 1:
                        #fo12--;
                        break;
                default:
                        break;
        }
        return (#fo12+#ba12 *#ca23 [#ba12])/- #fo12;
}
// note that functions are defined using a statement
int #ma11(int #aa11, #bb11)
        if (#aa11 .gt #bb11) return #aa11; else return #bb11;
```

Tokens

#	TYPE	VALUE	LINE NO.	LEX POSITION
0	RECORD_KW	record	1	0
1	ID	#po11	1	7
2	BR_OPEN	{	1	13
3	INT_T	int	2	17
4	ID	#xx11	2	21
5	COMMA	,	2	26
6	ID	#yy11	2	28
7	SEMICOLON	;	2	33
8	BR_CLOSE	}	3	36
9	RECORD_KW	record	5	41
10	ID	#li11	5	48
11	BR_OPEN	{	5	54
12	ID	#po11	6	58
13	ID	#xx11	6	64
14	COMMA	,	6	69
15	ID	#yy11	6	71
16	SEMICOLON	;	6	76
17	BR_CLOSE	}	7	79
18	INT_T	int	9	84
19	ID	#at11	9	88
20	PR_OPEN	(	9	94
21	INT_T	int	9	95
22	ID	#ba12	9	99
23	COMMA	,	9	104
24	ID	#ca23	9	106
25	BK_OPEN	[	9	111
26	BK_CLOSE	]	9	112
27	SEMICOLON	;	9	113
28	BOOL_T	bool	9	115
29	ID	#do43	9	120
30	COMMA	,	9	125
31	ID	#el32	9	127
32	SEMICOLON	;	9	132
33	INT_T	int	9	134
34	ID	#fo12	9	138
35	PR_CLOSE	)	9	143
36	BR_OPEN	{	10	146
37	INT_T	int	11	150
38	ID	#gn11	11	154
39	COMMA	,	11	159
40	ID	#ho12	11	161
41	BK_OPEN	[	11	166
42	NUMCONST	100	11	167
43	BK_CLOSE	]	11	170
44	SEMICOLON	;	11	171

45	REAL_T	real	12	175
46	ID	#el72	12	180
47	SEMICOLON	;	12	185
48	ID	#el72	13	189
49	EXP_OP	=	13	195
50	REALCONST	72.2	13	197
51	SEMICOLON	;	13	202
52	ID	#po11	14	206
53	ID	#aP11	14	212
54	SEMICOLON	;	14	217
55	ID	#li11	14	219
56	ID	#aL11	14	225
57	SEMICOLON	;	14	230
58	ID	#li11	15	234
59	ID	#tw33	15	240
60	BK_OPEN	[	15	245
61	NUMCONST	2	15	246
62	BK_CLOSE	]	15	247
63	SEMICOLON	;	15	248
64	ID	#aP11	16	256
65	DOT	.	16	261
66	ID	#xx11	16	262
67	EXP_OP	=	16	268
68	NUMCONST	666	16	270
69	SEMICOLON	;	16	273
70	ID	#aP11	16	275
71	DOT	.	16	280
72	ID	#yy11	16	281
73	EXP_OP	=	16	287
74	NUMCONST	667	16	289
75	SEMICOLON	;	16	292
76	ID	#aL11	17	300
77	DOT	.	17	305
78	ID	#xx11	17	306
79	DOT	.	17	311
80	ID	#xx11	17	312
81	EXP_OP	=	17	318
82	NUMCONST	1	17	320
83	SEMICOLON	;	17	321
84	ID	#aL11	17	323
85	DOT	.	17	328
86	ID	#xx11	17	329
87	DOT	.	17	334
88	ID	#yy11	17	335
89	EXP_OP	=	17	341
90	NUMCONST	2	17	343
91	SEMICOLON	;	17	344
92	ID	#aL11	17	346

93	DOT	.	17	351
94	ID	#yy11	17	352
95	DOT	.	17	357
96	ID	#xx11	17	358
97	EXP_OP	=	17	364
98	NUMCONST	3	17	366
99	SEMICOLON	;	17	367
100	ID	#aL11	17	369
101	DOT	.	17	374
102	ID	#yy11	17	375
103	DOT	.	17	380
104	ID	#yy11	17	381
105	EXP_OP	=	17	387
106	NUMCONST	4	17	389
107	SEMICOLON	;	17	390
108	ID	#tw33	18	398
109	BK_OPEN	[	18	403
110	NUMCONST	0	18	404
111	BK_CLOSE	]	18	405
112	DOT	.	18	406
113	ID	#xx11	18	407
114	DOT	.	18	412
115	ID	#xx11	18	413
116	EXP_OP	=	18	419
117	NUMCONST	42	18	421
118	SEMICOLON	;	18	423
119	ID	#tw33	18	425
120	BK_OPEN	[	18	430
121	NUMCONST	1	18	431
122	BK_CLOSE	]	18	432
123	DOT	.	18	433
124	ID	#yy11	18	434
125	DOT	.	18	439
126	ID	#xx11	18	440
127	EXP_OP	=	18	446
128	NUMCONST	43	18	448
129	SEMICOLON	;	18	450
130	ID	#gn11	19	458
131	EXP_OP	=	19	464
132	ID	#ho12	19	466
133	BK_OPEN	[	19	471
134	NUMCONST	2	19	472
135	BK_CLOSE	]	19	473
136	EXP_OP	=	19	475
137	NUMCONST	3	19	477
138	MATH_OP	*	19	478
139	MATH_OP	*	19	479
140	ID	#ca23	19	481

141	SEMICOLON	;	19	486
142	IF_KW	if	20	544
143	PR_OPEN	(	20	547
144	ID	#do43	20	548
145	AND_KW	and	20	554
146	ID	#el32	20	558
147	OR_KW	or	20	564
148	ID	#ba12	20	567
149	REL_OP	.gt	20	573
150	ID	#ca23	20	577
151	BK_OPEN	[	20	582
152	NUMCONST	3	20	583
153	BK_CLOSE	]	20	584
154	PR_CLOSE	)	20	585
155	ID	#do43	20	587
156	EXP_OP	=	20	593
157	NOT_KW	not	20	595
158	ID	#do43	20	599
159	SEMICOLON	;	20	604
160	ELSE_KW	else	21	612
161	ID	#fo12	21	617
162	MATH_OP	+	21	622
163	MATH_OP	+	21	623
164	SEMICOLON	;	21	624
165	IF_KW	if	22	632
166	PR_OPEN	(	22	635
167	ID	#ba12	22	636
168	REL_OP	.le	22	642
169	ID	#fo12	22	646
170	PR_CLOSE	)	22	651
171	BR_OPEN	{	22	653
172	WHILE_KW	while	23	665
173	PR_OPEN	(	23	671
174	ID	#do43	23	672
175	PR_CLOSE	)	23	677
176	BR_OPEN	{	23	679
177	STATIC_KW	static	24	696
178	INT_T	int	24	703
179	ID	#ho12	24	707
180	SEMICOLON	;	24	712
181	ID	#ho12	25	749
182	EXP_OP	=	25	755
183	ID	#fo12	25	757
184	SEMICOLON	;	25	762
185	ID	#do43	26	779
186	EXP_OP	=	26	785
187	ID	#fr77	26	787
188	PR_OPEN	(	26	792

189	ID	#fo12	26	793
190	MATH_OP	+	26	798
191	MATH_OP	+	26	799
192	COMMA	,	26	800
193	ID	#ca23	26	802
194	PR_CLOSE	)	26	807
195	REL_OP	.lt	26	809
196	NUMCONST	666	26	813
197	SEMICOLON	;	26	816
198	IF_KW	if	27	833
199	PR_OPEN	(	27	836
200	ID	#ho12	27	837
201	REL_OP	.gt	27	843
202	ID	#ba12	27	847
203	PR_CLOSE	)	27	852
204	BREAK_KW	break	27	854
205	SEMICOLON	;	27	859
206	ELSE_KW	else	28	875
207	IF_KW	if	28	880
208	PR_OPEN	(	28	883
209	ID	#fo12	28	884
210	REL_OP	.ne	28	890
211	NUMCONST	0	28	894
212	PR_CLOSE	)	28	895
213	ID	#fo12	28	897
214	MATH_OP	+	28	903
215	EXP_OP	=	28	904
216	NUMCONST	7	28	906
217	SEMICOLON	;	28	907
218	BR_CLOSE	}	29	919
219	BR_CLOSE	}	30	927
220	ID	#fo12	31	931
221	EXP_OP	=	31	937
222	MATH_OP	?	31	939
223	NUMCONST	5	31	940
224	SEMICOLON	;	31	941
225	SWITCH_KW	switch	32	945
226	PR_OPEN	(	32	952
227	ID	#fo12	32	953
228	PR_CLOSE	)	32	958
229	BR_OPEN	{	32	960
230	CASE_KW	case	33	965
231	NUMCONST	0	33	970
232	COLON	:	33	971
233	ID	#fo12	34	977
234	MATH_OP	+	34	982
235	MATH_OP	+	34	983
236	SEMICOLON	;	34	984

237	BREAK_KW	break	35	990
238	SEMICOLON	;	35	995
239	CASE_KW	case	36	1000
240	NUMCONST	1	36	1005
241	COLON	:	36	1006
242	ID	#fo12	37	1012
243	MATH_OP	-	37	1017
244	MATH_OP	-	37	1018
245	SEMICOLON	;	37	1019
246	BREAK_KW	break	38	1025
247	SEMICOLON	;	38	1030
248	DEFAULT_KW	default	39	1035
249	COLON	:	39	1042
250	BREAK_KW	break	40	1048
251	SEMICOLON	;	40	1053
252	BR_CLOSE	}	41	1057
253	RETURN_KW	return	42	1065
254	PR_OPEN	(	42	1072
255	ID	#fo12	42	1073
256	MATH_OP	+	42	1078
257	ID	#ba12	42	1079
258	MATH_OP	*	42	1085
259	ID	#ca23	42	1086
260	BK_OPEN	[	42	1092
261	ID	#ba12	42	1093
262	BK_CLOSE	]	42	1098
263	PR_CLOSE	)	42	1099
264	MATH_OP	/	42	1100
265	MATH_OP	-	42	1101
266	ID	#fo12	42	1103
267	SEMICOLON	;	42	1108
268	BR_CLOSE	}	43	1111
269	INT_T	int	45	1168
270	ID	#ma11	45	1172
271	PR_OPEN	(	45	1177
272	INT_T	int	45	1178
273	ID	#aa11	45	1182
274	COMMA	,	45	1187
275	ID	#bb11	45	1189
276	PR_CLOSE	)	45	1194
277	IF_KW	if	46	1198
278	PR_OPEN	(	46	1201
279	ID	#aa11	46	1202
280	REL_OP	.gt	46	1208
281	ID	#bb11	46	1212
282	PR_CLOSE	)	46	1217
283	RETURN_KW	return	46	1219
284	ID	#aa11	46	1226

285	SEMICOLON	;	46	1231
286	ELSE_KW	else	46	1233
287	RETURN_KW	return	46	1238
288	ID	#bb11	46	1245
289	SEMICOLON	;	46	1250