Rachel Chiang
CS 411-01
Project #1

TEST #1

Input givenOddCases.lex

```
// keywords that are reserved words. There should be 20.
boolean break class double else extends false for if implements
int interface newarray println readln return string true void while

// case-sensitive identifiers
IF hello Hello

// ifintvoid versus if(23void
ifintvoid
if(23void

// integer constants
0x0
0X12aE
8
012

// valid double constants
0.12
12.
12.2E+2
12.E+2

// invalid double constants
.12
1.2E
.12E+2

// valid string
"hello"

//invalid string
"good
bye"
```

Output

```
boolean break class double else extends booleanconstant for if implements
int interface newarray println readln return string booleanconstant void while


id id id


id
if leftparen intconstant void


intconstant
intconstant
intconstant
intconstant


doubleconstant
doubleconstant
doubleconstant
doubleconstant
```

```
period intconstant
doubleconstant id
period intconstant id add intconstant


stringconstant


stringconstant

switch:
     A     B     C     D     E     F     G     H     I     J     K     L     M
    -1    -1    -1    -1   131    -1    -1   116   109    -1    -1    -1    -1

     N     O     P     Q     R     S     T     U     V     W     X     Y     Z
    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1

     a     b     c     d     e     f     g     h     i     j     k     l     m
    -1     0    12    17    23    34    -1   111    42    -1    -1    -1    -1

     n     o     p     q     r     s     t     u     v     w     x     y     z
    64    -1    72    -1    79    90    96    -1   100   104    -1    -1    -1


index:
symbol:
next:
       0       1       2       3       4       5       6       7       8       9
       o       o       l       e       a       n       $       r       e       a
       7

      10      11      12      13      14      15      16      17      18      19
       k       $       l       a       s       s       $       o       u       b


      20      21      22      23      24      25      26      27      28      29
       l       e       $       l       s       e       $       x       t       e
                               27

      30      31      32      33      34      35      36      37      38      39
       n       d       s       $       a       l       s       e       $       o
                                       39

      40      41      42      43      44      45      46      47      48      49
       r       $       f       $       m       p       l       e       m       e
                       44     121      54

      50      51      52      53      54      55      56      57      58      59
       n       t       s       $       n       t       $       e       r       f
                                               57

      60      61      62      63      64      65      66      67      68      69
       a       c       e       $       e       w       a       r       r       a


      70      71      72      73      74      75      76      77      78      79
       y       $       r       i       n       t       l       n       $       e


      80      81      82      83      84      85      86      87      88      89
       a       d       l       n       $       t       u       r       n       $
```

```
 85

 90   91   92   93   94   95   96   97   98   99
  t    r    i    n    g    $    r    u    e    $


100  101  102  103  104  105  106  107  108  109
  o    i    d    $    h    i    l    e    $    F
                 130

110  111  112  113  114  115  116  117  118  119
  $    e    l    l    o    $    e    l    l    o


120  121  122  123  124  125  126  127  128  129
  $    i    n    t    v    o    i    d    $    $
       129

130  131  132
  $    $    $
       132
```

TEST #2

Input givenToy.lex

```
int fact (int x) {
// recursive factorial function
    if (x>1) return x * fact(x-1);
    else return 1;
}
void main () {
/* Winter Quarter 2015
CS 411 project #1
A lexical analyzer */
    int x;
    int total;
    println ("factorial of 10 is ", fact (10), " from the recursive function");
    total = 1; x = 1;
    for ( ; x<=10; ) { total = total * x; x = x + 1; }
    println ("iterative result of 10! is ", total);
}
class cs411 {
    int Funny;
    double funny;
    boolean flag;
    string s;
    int [] a;
    flag = true;
    Funny = 0X89aB; funny = 123456E+7;
    s = "hello world";
    while (x = (Funny/10) <0) println (s, " have fun !");
    a = newarray (20, int);
}
```

Output

```
int id leftparen int id rightparen leftbrace

if leftparen id greaterthan intconstant rightparen return id mult id leftparen id sub intconstant
rightparen semicolon
else return intconstant semicolon
rightbrace
```

void id leftparen rightparen leftbrace

int id semicolon
int id semicolon
println leftparen stringconstant comma id leftparen intconstant rightparen comma stringconstant rightparen semicolon
id assignop intconstant semicolon id assignop intconstant semicolon
for leftparen semicolon id lessequal intconstant semicolon rightparen leftbrace id assignop id mult id semicolon id assignop id add intconstant semicolon rightbrace
println leftparen stringconstant comma id rightparen semicolon
rightbrace
class id leftbrace
int id semicolon
double id semicolon
boolean id semicolon
string id semicolon
int leftbracket rightbracket id semicolon
id assignop booleanconstant semicolon
id assignop intconstant semicolon id assignop intconstant id add intconstant semicolon
id assignop stringconstant semicolon
while leftparen id assignop leftparen id div intconstant rightparen lessthan intconstant rightparen println leftparen id comma stringconstant rightparen semicolon
id assignop newarray leftparen intconstant comma int rightparen semicolon
rightbrace
switch:

A	B	C	D	E	F	G	H	I	J	K	L	M
-1	-1	-1	-1	117	74	-1	-1	-1	-1	-1	-1	-1

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

a	b	c	d	e	f	g	h	i	j	k	l	m
109	90	63	79	21	3	-1	-1	0	-1	-1	-1	30

n	o	p	q	r	s	t	u	v	w	x	y	z
129	-1	42	-1	12	101	37	-1	26	119	8	-1	-1

index:
symbol:
next:

0	1	2	3	4	5	6	7	8	9
n	t	$	a	c	t	$	$	$	f
9		7	52			19	34	11	

10	11	12	13	14	15	16	17	18	19
$	$	e	t	u	r	n	$	$	$
	18						25	20	49

20	21	22	23	24	25	26	27	28	29
$	l	s	e	$	$	o	i	d	$
35									

30	31	32	33	34	35	36	37	38	39
a	i	n	$	$	$	$	o	t	a
				36	51	73	111		

40	41	42	43	44	45	46	47	48	49
l	$	r	i	n	t	l	n	$	$
	50							61	

50	51	52	53	54	55	56	57	58	59
$	$	o	r	$	$	$	$	$	$
56	55	85			58	57	62	59	60

60	61	62	63	64	65	66	67	68	69
$	$	$	l	a	s	s	$	s	4
124	126		68						

70	71	72	73	74	75	76	77	78	79
1	1	$	$	u	n	n	y	$	o
			108					115	

80	81	82	83	84	85	86	87	88	89
u	b	l	e	$	u	n	n	y	$
					97				116

90	91	92	93	94	95	96	97	98	99
o	o	l	e	a	n	$	l	a	g

100	101	102	103	104	105	106	107	108	109
$	t	r	i	n	g	$	$	$	$
110	107						118	137	128

110	111	112	113	114	115	116	117	118	119
$	r	u	e	$	$	$	$	$	h
					125			127	

120	121	122	123	124	125	126	127	128	129
i	l	e	$	$	$	$	$	$	e

130	131	132	133	134	135	136	137
w	a	r	r	a	y	$	$

TEST #3

Input givenTrie.lex

```
// This should output the same trie structure as was given in the project
// specifications. I am uncertain about whether to store true or false
// but I have chosen to store them anyway
boolean break class double else extends false for if implements int interface
```

Output

```
boolean break class double else extends booleanconstant for if implements int interface

switch:
```

A	B	C	D	E	F	G	H	I	J	K	L	M
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

a	b	c	d	e	f	g	h	i	j	k	l	m
-1	0	12	17	23	34	-1	-1	42	-1	-1	-1	-1

n	o	p	q	r	s	t	u	v	w	x	y	z
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

```
index:
symbol:
next:
       0       1       2       3       4       5       6       7       8       9
       o       o       l       e       a       n       $       r       e       a
       7

      10      11      12      13      14      15      16      17      18      19
       k       $       l       a       s       s       $       o       u       b


      20      21      22      23      24      25      26      27      28      29
       l       e       $       l       s       e       $       x       t       e
                               27

      30      31      32      33      34      35      36      37      38      39
       n       d       s       $       a       l       s       e       $       o
                                       39

      40      41      42      43      44      45      46      47      48      49
       r       $       f       $       m       p       l       e       m       e
                      44              54

      50      51      52      53      54      55      56      57      58      59
       n       t       s       $       n       t       $       e       r       f
                                                       57

      60      61      62      63
       a       c       e       $
```

TEST #4

Input sillyCase.lex

```
classes Bad implement other
{
    printf("oh");
}
```

Output

```
id id id id
leftbrace
id leftparen stringconstant rightparen semicolon
rightbrace

switch:
     A     B     C     D     E     F     G     H     I     J     K     L     M
    -1     7    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1

     N     O     P     Q     R     S     T     U     V     W     X     Y     Z
    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1

     a     b     c     d     e     f     g     h     i     j     k     l     m
    -1    -1     0    -1    -1    -1    -1    -1    10    -1    -1    -1    -1

     n     o     p     q     r     s     t     u     v     w     x     y     z
    -1    19    24    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1


index:
symbol:
```

```
next:
       0       1       2       3       4       5       6       7       8       9
       l       a       s       s       e       s       $       a       d       $


      10      11      12      13      14      15      16      17      18      19
       m       p       l       e       m       e       n       t       $       t


      20      21      22      23      24      25      26      27      28      29
       h       e       r       $       r       i       n       t       f       $
```

TEST #5

Input simpleToy.lex

```
boolean checkRange(int n)
{
    // checks if n is in the range [0,100]
    if (n >= 0 && n <= 100)
        return true;
    return false;
}

void main()
{
    // a weirdly named string
    string String = "Hello, world!";
    // This prints Hello, world!
    println(String);

    /* I'm not really sure what the purpose of this is */
    int i = 0;
    for (int i = 0; i < 200; i = i + 2)
    {
        if (checkRange(i))
        {
            printf("You are within range.");
        }
    }
}
```

Output

```
boolean id leftparen int id rightparen
leftbrace

if leftparen id greaterequal intconstant and id lessequal intconstant rightparen
return booleanconstant semicolon
return booleanconstant semicolon
rightbrace

void id leftparen rightparen
leftbrace

string id assignop stringconstant semicolon

println leftparen id rightparen semicolon


int id assignop intconstant semicolon
```

```
for leftparen int id assignop intconstant semicolon id lessthan intconstant semicolon id assignop
id add intconstant rightparen
leftbrace
if leftparen id leftparen id rightparen rightparen
leftbrace
id leftparen stringconstant rightparen semicolon
rightbrace
rightbrace
rightbrace
```

switch:

A	B	C	D	E	F	G	H	I	J	K	L	M
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
-1	-1	-1	-1	-1	55	-1	-1	-1	-1	-1	-1	-1

a	b	c	d	e	f	g	h	i	j	k	l	m
-1	0	7	-1	-1	36	-1	-1	17	-1	-1	-1	45

n	o	p	q	r	s	t	u	v	w	x	y	z
20	-1	61	-1	25	49	31	-1	41	-1	-1	-1	-1

index:
symbol:
next:

0	1	2	3	4	5	6	7	8	9
o	o	l	e	a	n	$	h	e	c

10	11	12	13	14	15	16	17	18	19
k	R	a	n	g	e	$	n	t	$
						80	21		69

20	21	22	23	24	25	26	27	28	29
$	f	$	$	$	e	t	u	r	n
23	70	79	24						

30	31	32	33	34	35	36	37	38	39
$	r	u	e	$	$	a	l	s	e
35						71			

40	41	42	43	44	45	46	47	48	49
$	o	i	d	$	a	i	n	$	t

50	51	52	53	54	55	56	57	58	59
r	i	n	g	$	t	r	i	n	g

60	61	62	63	64	65	66	67	68	69
$	r	i	n	t	l	n	$	$	$
68					82				74

70	71	72	73	74	75	76	77	78	79
$	o	r	$	$	$	$	$	$	$
75					76	77	78	81	

80	81	82	83
$	$	f	$

TEST #6

Note: I’m putting these in columns and making them small because it’s a massive waste of paper

Input growTest.lex

// Just some random strings	BtJCvFlSimjKSeA	OOpsaYJFQveQzzS	cxKuqmAwGDgtOBB
LFwnmeYxNitVmDZ	vqTbMuIjyMLEjQm	hUMbYtRXCkbKsJE	qMKjsFRkczoLISW
sVbBsqvbhsmHNYJ	IHjdaMqXruMPIsO	RpNHiVytDsEIgSG	TGmyWvNrNrhoVUh
lUZHqHcyrulQPVR	soWySBLqSufyyKD	byRbeyJqINMOZxY	LyeuVzMmxcNbIfo
bmwWcUDdDaaRmlF	BvcuNLSzBBgLtLr	dVuGaqaARQiQItV	EVdPJayqAJKZAEd
FHQprmxRwffBIgR	sbXrvCECrIgVXOY	vtTCIvJqstaEqtR	ZFdIJPVPvVepdjT
VqmsFAZtMYrpYtH	toOglDEXhkdXXxP	dncbVuarAPHZDjI	cMBaWftcAqfjVJk
EYGfCbnjlbqcWRS	duNlimKTYPivavZ	QIgNKbSBryGYpiT	gpbrRAjAbnVEgBE
rgYeBLHrnerxDkM	NZeqMJYdLCUTXOM	ykgdcxRueZKeoCH	
bGKRKMvrtxemEDy	rTwkMAtqefnYVFP	MOLWgKiGNzBsROx	
SCfmJyfvQHuwiwY	lpUDpNmNkXlmqMD	GdRixRUIOGLmSpP	

Output

id	id	id	id
id	id	id	id
id	id	id	id
id	id	id	id
id	id	id	id
id	id	id	id
id	id	id	id
id	id	id	id
id	id	id	id
id	id	id	id

switch:

A	B	C	D	E	F	G	H	I	J	K	L	M
-1	150	-1	-1	90	60	465	-1	180	-1	-1	0	450

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
270	315	-1	420	345	135	510	-1	75	-1	-1	-1	555

a	b	c	d	e	f	g	h	i	j	k	l	m
-1	45	480	255	-1	-1	585	330	-1	-1	-1	30	-1

n	o	p	q	r	s	t	u	v	w	x	y	z
-1	-1	-1	495	105	15	240	-1	165	-1	-1	435	-1

index:	0	1	2	3	4	5	6	7	8	9
symbol:	F	w	n	m	e	Y	x	N	i	t
next:	525									

10	11	12	13	14	15	16	17	18	19
V	m	D	Z	$	V	b	B	s	q
					195				

20	21	22	23	24	25	26	27	28	29
v	b	h	s	m	H	N	Y	J	$

30	31	32	33	34	35	36	37	38	39
U	Z	H	q	H	c	y	r	u	l
300									

40	41	42	43	44	45	46	47	48	49
Q	P	V	R	$	m	w	W	c	U
					120				

50	51	52	53	54	55	56	57	58	59
D	d	D	a	a	R	m	l	F	$

60	61	62	63	64	65	66	67	68	69
H	Q	p	r	m	x	R	w	f	f

70	71	72	73	74	75	76	77	78	79
B	I	g	R	$	q	m	s	F	A

80	81	82	83	84	85	86	87	88	89
Z	t	M	Y	r	p	Y	t	H	$

90	91	92	93	94	95	96	97	98	99
Y	G	f	C	b	n	j	l	b	q
540									

100	101	102	103	104	105	106	107	108	109
c	W	R	S	$	g	Y	e	B	L
					285				

110	111	112	113	114	115	116	117	118	119
H	r	n	e	r	x	D	k	M	$

120	121	122	123	124	125	126	127	128	129
G	K	R	K	M	v	r	t	x	e
360									

130	131	132	133	134	135	136	137	138	139
m	E	D	y	$	C	f	m	J	y

140	141	142	143	144	145	146	147	148	149
f	v	Q	H	u	w	i	w	Y	$

150	151	152	153	154	155	156	157	158	159
t	J	C	v	F	l	S	i	m	j
210									

160	161	162	163	164	165	166	167	168	169
K	S	e	A	$	q	T	b	M	u
					390				

170	171	172	173	174	175	176	177	178	179
I	j	y	M	L	E	j	Q	m	$

180	181	182	183	184	185	186	187	188	189
H	j	d	a	M	q	X	r	u	M

190	191	192	193	194	195	196	197	198	199
P	I	s	O	$	o	W	y	S	B
					225				

200	201	202	203	204	205	206	207	208	209
L	q	S	u	f	y	y	K	D	$

210	211	212	213	214	215	216	217	218	219
v	c	u	N	L	S	z	B	B	g

220	221	222	223	224	225	226	227	228	229

L	t	L	r	$	b	X	r	v	C
230	231	232	233	234	235	236	237	238	239
E	C	r	I	g	V	X	O	Y	$
240	241	242	243	244	245	246	247	248	249
o	O	g	l	D	E	X	h	k	d
250	251	252	253	254	255	256	257	258	259
X	X	x	P	$	u	N	l	i	m
375									
260	261	262	263	264	265	266	267	268	269
K	T	Y	P	i	v	a	v	Z	$
270	271	272	273	274	275	276	277	278	279
Z	e	q	M	J	Y	d	L	C	U
280	281	282	283	284	285	286	287	288	289
T	X	O	M	$	T	w	k	M	A
290	291	292	293	294	295	296	297	298	299
t	q	e	f	n	Y	V	F	P	$
300	301	302	303	304	305	306	307	308	309
p	U	D	p	N	m	N	k	X	l
310	311	312	313	314	315	316	317	318	319
m	q	M	D	$	O	p	s	a	Y
320	321	322	323	324	325	326	327	328	329
J	F	Q	v	e	Q	z	z	S	$
330	331	332	333	334	335	336	337	338	339
U	M	b	Y	t	R	X	C	k	b
340	341	342	343	344	345	346	347	348	349
K	s	J	E	$	p	N	H	i	V
350	351	352	353	354	355	356	357	358	359
y	t	D	s	E	I	g	S	G	$
360	361	362	363	364	365	366	367	368	369
y	R	b	e	y	J	q	I	N	M
370	371	372	373	374	375	376	377	378	379
O	Z	x	Y	$	V	u	G	a	q
405									
380	381	382	383	384	385	386	387	388	389
a	A	R	Q	i	Q	I	t	V	$
390	391	392	393	394	395	396	397	398	399
t	T	C	I	v	J	q	s	t	a
400	401	402	403	404	405	406	407	408	409
E	q	t	R	$	n	c	b	V	u
410	411	412	413	414	415	416	417	418	419
a	r	A	P	H	Z	D	j	I	$

420	421	422	423	424	425	426	427	428	429
I	g	N	K	b	S	B	r	y	G
430	431	432	433	434	435	436	437	438	439
Y	p	i	T	$	k	g	d	c	x
440	441	442	443	444	445	446	447	448	449
R	u	e	Z	K	e	o	C	H	$
450	451	452	453	454	455	456	457	458	459
O	L	W	g	K	i	G	N	z	B
460	461	462	463	464	465	466	467	468	469
s	R	O	x	$	d	R	i	x	R
470	471	472	473	474	475	476	477	478	479
U	I	O	G	L	m	S	p	P	$
480	481	482	483	484	485	486	487	488	489
x	K	u	q	m	A	w	G	D	g
570									
490	491	492	493	494	495	496	497	498	499
t	O	B	B	$	M	K	j	s	F
500	501	502	503	504	505	506	507	508	509
R	k	c	z	o	L	I	S	W	$
510	511	512	513	514	515	516	517	518	519
G	m	y	W	v	N	r	N	r	h
520	521	522	523	524	525	526	527	528	529
o	V	U	h	$	y	e	u	V	z
530	531	532	533	534	535	536	537	538	539
M	m	x	c	N	b	I	f	o	$
540	541	542	543	544	545	546	547	548	549
V	d	P	J	a	y	q	A	J	K
550	551	552	553	554	555	556	557	558	559
Z	A	E	d	$	F	d	I	J	P
560	561	562	563	564	565	566	567	568	569
V	P	v	V	e	p	d	j	T	$
570	571	572	573	574	575	576	577	578	579
M	B	a	W	f	t	c	A	q	f
580	581	582	583	584	585	586	587	588	589
j	V	J	k	$	p	b	r	R	A
590	591	592	593	594	595	596	597	598	599
j	A	b	n	V	E	g	B	E	$

TEST #7

Input bad.lex

```
// This should output id mult id leftbracket id comma comma id

fwoie*fiowejfosdp[fjwe0c,,cowo
```

Output

```
id mult id leftbracket id comma comma id

switch:
```

A	B	C	D	E	F	G	H	I	J	K	L	M
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

a	b	c	d	e	f	g	h	i	j	k	l	m
-1	-1	22	-1	-1	0	-1	-1	-1	-1	-1	-1	-1

n	o	p	q	r	s	t	u	v	w	x	y	z

```
    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1


index:
symbol:
next:
       0       1       2       3       4       5       6       7       8       9
       w       o       i       e       $       i       o       w       e       j
       5                                       16

      10      11      12      13      14      15      16      17      18      19
       f       o       s       d       p       $       j       w       e       0


      20      21      22      23      24      25
       c       $       o       w       o       $
```

TEST #8

Input simpleIDs.lex

```
/* These are examples we did in class. */
awt awthave
cry
an
and
am
awtha
```

Output

```
id id
id
id
id
id
id

switch:
     A     B     C     D     E     F     G     H     I     J     K     L     M
    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1

     N     O     P     Q     R     S     T     U     V     W     X     Y     Z
    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1

     a     b     c     d     e     f     g     h     i     j     k     l     m
     0    -1     8    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1

     n     o     p     q     r     s     t     u     v     w     x     y     z
    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1    -1


index:
symbol:
next:
       0       1       2       3       4       5       6       7       8       9
       w       t       $       h       a       v       e       $       r       y
      11               3                       17

      10      11      12      13      14      15      16      17
       $       n       $       d       $       m       $       $
               15      13
```