Sharpie

Ali Babayev - 21600475 (section 1)

Nurlan Farzaliyev - 21503756 (section 1)

Mahammad Shirinov - 21603176 (section 2)

Sharpie

BNF

```
<set> -> [<set_elements>]
<set_elements> -> <set_element>
                | <set_elements>, <set_element>
<set_element> -> <data_type>
<identifier> -> <letter>
                | <identifier><letter>
                | <identifier><digit>
<letter> -> a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
           A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|K|R|S|T|U|V|W|X|Y|Z
<digit> -> 0|<positive_digit>
<positive_digit> -> 1|2|3|4|5|6|7|8|9
<num> -> <digit>|<num><digit>
<data_type> -> <primitive_data_type> | <set> | <string>
<primitive_data_type> -> <int>
                | <double>
                | <boolean>
<int> -> <sign>?<positive_digit>|<int><digit>
<double> -> <sign>?<positive_digit><dot_sign><num>|<double>
<boolean> -> true|false
<string> -> #<char_array>#
<char_array> -> <char> | <char_array><char>
<sign> -> -|+
<dot_sign> -> .
<set_composition> -> <set><set_op><set>
<set_op> -> & | ^ | -
<set_inclusion> -> <set><inclusion_symbol><set>
<inclusion_symbol> -> < | >
<assignment_operator> -> =
<while_loop> -> while(<boolean_expression>)<code_block>
<code_block> -> {<statements>}
<boolean_expression> ->
<primitive_data_type><logical_operand><primitive_data_type>
                | <set_inclusion>
<logical_operand> -> <equals> | <nequals>
<statements> -> <statement>; | <statements><statement>
<statement> -> <identifier> <assignment_operator> <expression>
                | <def_var>
                | <oxu_statement>
```

```
                | <yaz_statement>
                | <function_def>
                | <function_call>
                | <while_loop>
                | <if_statement>
<type> -> int | double | boolean | set | string

<expression> -> <set>
                | <data_type>
                | <expression><assignment_operator><data_type>
                | <set_composition>
                | <set_inclusion>

<oxu_statement> -> oxu(<identifier>)
<yaz_statement> -> yaz(<identifier>)
<if_statement> -> if(<boolean_expression>)<code_block>
                |if(<boolean_expression>)<code_block>else<code_block>

<function_def> -> <return_type> <identifier> (<def_var_list>)
<code_block>
<return_type> -> <type> | void
<def_var_list> -> void
                | <def_var>
                | <def_var_list>, <def_var>
<def_var> -> <type> <identifier>
<function_call> -> <identifier>(<parameter_list>)
<parameter_list> -> <identifier>
                | <parameter_list>, <identifier>

<keyword> -> void | true | false | int | double | boolean | set | oxu
| yaz | string
<comment> -> //<identifier> | <comment> <identifier>
```

The elements of our language:

`<set>`: a set is a collection of ‘things’, which can include integers, doubles, strings, booleans or other sets. In syntax, it is a comma separated list of identifiers that refer to these ‘data types’, enclosed in square brackets.

<set_inclusion>: this captures set inclusion statements, returning a boolean value of true or false. Set inclusion is denoted by characters '<<' and '>>', where X<<Y (same as Y>>X) means "X is a subset of Y". For example, if set A is a subset of B, the value of A<<B will be true.
<set_composition>: this is an operation between two or more sets, returning a set of intersection or union, or difference of the argument sets.
<identifier>: a sequence of characters that is used to identify variables, data types, or functions. They are basically used as names.
yaz and oxu keywords: names of functions for output and input, respectively
<type>: is one of the keywords int, double, boolean, set, string that are used to define the types of data.
<data_type>: this is used to represent data; they can be primitive data types (integer, double and boolean values) or strings and sets. Unlike <type>s, these refer to the actual values of data_types, and not to their labels or locations.
<char>: any ASCII character except for '#'.
<string>: strings are always surrounded by a pair of '#'.
<comment>: single line of characters ignored by the compiler, starting with '//'

Lex file

```
%option main
integer         [0-9]+
double          [+-]?[0-9]*(\.)?[0-9]+
letter          [a-zA-Z]
identifier      [a-zA-Z][a-zA-Z0-9]*
string          #[a-zA-Z0-9_.,$@!?>=< |:]*#
comment         \/\/[a-zA-Z0-9_.,$@!?>=< |:]*
endofstatement  \;
set             \[[a-zA-Z][a-zA-Z0-9]*(\,\ [a-zA-Z][a-zA-Z0-9]*)*\]
lcb             \{
rcb             \}
lp              \(
rp              \)
union           \&
intersection    \^
complement      \~
subsetof        \<\<
superof         \>\>
setvalueto      \=
equals          \==
nequals         \!\=
boolean         true|false
print           yaz
read            oxu
type            int|boolean|double|set|string
while           while
typeint         int
typedouble      double
typeboolean     boolean
typeset         set
typestring      string
typevoid        void
if              if
%%
\n              printf("\n");
{comment}       printf(" A COMMENT ");
{if}            printf(" IF ");
{lp}            printf(" LP ");
```

```
{rp}            printf(" RP ");
{lcb}           printf(" LCB ");
{rcb}           printf(" RCB ");
{set}           printf(" SET ");
{integer}       printf(" INTEGER ");
{while}         printf(" WHILE ");
{double}        printf(" DOUBLE ");
{string}        printf(" STRING ");
{union}         printf(" UNION WITH ");
{intersection}  printf(" INTERSECTION WITH ");
{complement}    printf(" COMPLEMENT WRT ");
{endofstatement} printf(" SEMIC ");
{subsetof}      printf(" SUBSET OF ");
{superof}       printf(" SUPERSET OF ");
{setvalueto}    printf(" IS SET TO ");
{equals}        printf(" EQUALS ");
{nequals}            printf(" NEQUALS ");
{print}         printf(" PRINT ");
{read}          printf(" READ INPUT ");
{typeint}       printf(" TYPE INTEGER ");
{typedouble}    printf(" TYPE DOUBLE ");
{typeset}       printf(" TYPE SET ");
{typestring}    printf(" TYPE STRING ");
{typeboolean}   printf(" TYPE BOOLEAN ");
{typevoid}      printf(" TYPE VOID ");
{identifier}    printf(" IDENTIFIER ");
```

Example Program

```
int a = 5;
double b = 7.0;
boolean c = true;
string str = #Salam salam!>=<#;
//this is the comment !>=<.
set setOfNumbers = [1,2,3];
set setOfsets = [[1,2,3],[1],[2],[3]];
yaz(setOfset);
yaz(#It was set of sets#);
set intersectionOfTwoSets = [1,2] ^ [3,4];
```

```
createSet(setOfNumbers, setOfsets);
void createSet(set set1, set set2) { set mergeSets = set1 &
set2;}
boolean isSubset;
if(setOfNumbers << setOfsets) {subset = true;}
if(setOfNumbers >> setOfsets) {subset = false;}
string inputStr;
oxu(inputStr);
int counter = 5;
while(counter != 0) {
    yaz(inputStr);
}
```