INTRODUCTION AND GENERAL SEMANTICS

HL is a domain-specific language focusing on the manipulation of ordered sets of integers, including integer intervals. HL is a scoped language and therefore variables need to be declared before they are used in their scope. However, HL has implicit typing: the type of a variable is determined from the first character of its name and therefore the variable's type does not need to be specified in its declaration. The HL types are:

- Numbers (NUM) are all integers (elements of $\mathbb{Z}$). There are no other numerical types in HL such as decimals or floats. Number variables must start with a lower case letter.
- Sets (SET) are ordered list of numbers. All set variables must start with an upper case letter.
- Booleans (BOOL) can have one of two values: #1 (true) or #0 (false). All Boolean variables must start with a pound sign ("#") followed by a letter.

Note that HL does not contain a string type and therefore there are no string variables. However strings constants are used in print statements in HL.

HL has operators which are overloaded to work with values of type NUM, BOOL and SET. This document describes the semantics of these operations.

GENERAL EVALUATION SEMANTICS

- All expressions in HL should evaluate to a specific value of one of the three HL types.
- Numbers, Booleans, and set literals (constants) evaluate to their value. When a set is evaluated, all its elements are evaluated.
- An identifier will evaluate to its value in the scope in which the evaluation takes place. This value will be a literal of one of HL's three types.
- When statements other than functions calls (which are also expressions) are evaluated, they do not result in a value that can be manipulated. Instead they are executed. Most HL statements have the same semantics as in other common languages. These are discussed further in the next assignment.

SPECIFIC SETS SEMANTICS

Sets in HL can only contain elements of HL which are numbers. Two other issues must taken into consideration during set operations and set comparisons.

- Like sets in set theory, in an HL set, duplicated elements are considered to be the same element. E.g. {1,1} is the same as {1}. For this reason, when a set is evaluated, it must also be simplified to remove all duplicates. For example, the following program should print the set {1} and not the set {1,1}:

```
a = 1;        b = 1;        print {a,b};
```

- Unlike sets in set theory, HL SETs always store and display their elements in increasing numerical order. However, since it is not always possible to guarantee that sets will be specified in order, the onus is on the HL evaluator to sort SET elements. For example, in an HL program, the following snipet of code will print the set {1,3} twice and not {3,1} twice:

```
a = 1;        b = 3;        print {b,a};    print {3,1};
```

INTERVALS

HL has a special notation for intervals of numbers: [x,y] is the HL set of all numbers between x and y inclusively. For example:

- [-3,8] is the set {-3,-2,-1,0,1,2,3,4,5,6,7,8}
- [-2,-2] is the set {-2}
- [8,-3] is the null set {} because 8>-3

Because all the elements of HL sets are integers, the open interval notation used with real numbers in mathematics is not necessary in HL. For example,]-3,8[can simply be represented as [-2,7] in HL.

VALID BINARY AND UNARY OPERATIONS: EMPTY CELLS ARE INVALID OPERATIONS.

Op1	Op2	Result	Op1 + Op2	Op1 – Op2	Op1 *Op2	Op1 /Op2	Op1 % Op2
NUM	NUM	NUM	Arithmetic addition	Arithmetic subtraction	Arithmetic multiplication	Quotient of integer division	Remainder of integer division
SET	SET	SET	Set union	Set difference: the resulting set consists of all the elements of op1 which are not in op2	Set intersection		

Op1	+ Op1	- Op1	| Op1 |
NUM	Op1	0 - Op1	Absolute value of Op1
SET	Op1	The unary – operator is applied to all the elements of Op1	Number of elements in Op1

COMPARISONS: EMPTY CELLS ARE INVALID COMPARISONS

op1	NUM	SET	NUM
op2	NUM	SET	SET
<	#1 iff op1 < op2	#1 iff op1 ⊂ op2	
<=	#1 iff op1 ≤ op2	#1 iff op1 ⊆ op2	
>	#1 iff op1 > op2	#1 iff op1 ⊃ op2	
>=	#1 iff op1 ≥ op2	#1 iff op1 ⊇ op2	
==	#1 iff op1 = op2	#1 iff op1 = op2	
!=	#1 iff op1 ≠ op2	#1 iff op1 ≠ op2	
=in			#1 iff Op1 ∈ Op2
!in			#1 iff Op1 ∉ Op2

BOOLEAN OPERATORS

- The three Boolean operators !, &, | work like in other computer languages
- Boolean operators & and | are short-circuit operators.

RULES OF EVALUATION

As explained in the previous assignment, HL operations are left associative, and they follow these precedence rules, which have been integrated into the HL grammar and ASTs:

Highest	( )	* / %	+ -	comparisons	!	&	|	Lowest

ADDITIONAL SEMANTICS

All semantics involving identifiers will be provided in the next assignment. This includes sets defined with set formers, and most statements: declarations, assignments, loops, function calls other than the builtin card just defined, and returns..
Scoping and binding rules will also be provided at that time.