FILE: LEXIC.TXT

Alphabet:

a) Upper (A-Z) and lower (a-z) case letters of the English alphabet

b) Special symbols (defined below as spec\_symbol)

c) Decimal digits (0-9)

Lexic:

a) Special symbols, representing:

- operators: + - \* / % = < <= == >= > != and or not

- separators: () [] {} , ; space #

- reserved words: int string print input if elif else while

b) Identifiers:

- A sequence of letters, digits and underscores, starting with a letter:

identifier = letter{letter | digit | "\_"}

letter = "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"

digit = "0" | "1" | ... | "9"

c) Constants:

1) int:

const\_int = 0 | ["-"]nz\_digit{"0" | nzdigit}

nz\_digit = "1" | ... | "9"

2) string:

const\_string = “\““ {letter | spec\_symbol | digit} “\””

spec\_symbol = "$" | "%" | ... | "&" (\* Any special symbols including separators \*)

FILE: TOKENS.IN

Tokens:

+ - \* / % = < <= == >= > != and or not () [] {} , ; space # int string print input if elif else while

FILE: SYNTAX.IN

Syntax:

program = "{" stmtlist "}"

stmtlist = stmt\_full{stmt\_full}

stmt\_full = cmpdstmt | simplestmt

simplestmt = (definestmt | assignstmt | printstmt | inputstmt) “;”

cmpdstmt = (if | while)

ifstmt = "if" expression "{" stmtlist "}"

whilestmt = "while" expression "{" stmtlist "}"

definestmt = type (identifier\_advanced | assignstmt)

assignstmt = identifier\_advanced "=" term

printstmt = "print” “(" term ")"

inputstmt = "input” “(" type ")"

expression = "(" condition\_composite ")"

condition\_composite = condition\_advanced[logical\_binary\_op (condition\_advanced|condition\_composite)]

condition\_advanced = ["not"]condition\_simple

condition\_simple = ["("] term relational\_op term [")"]

logical\_binary\_op = "and" | "or"

algebric\_op = "+" | "-" | "\*" | "/" | "%"

relational\_op = "=" | "=" | "<" | "<=" | "==" | ">=" | ">" | "!="

type = "int" | "string"

term = factor | “(“ factor algebric\_op (factor|term) ")"

identifier\_advanced = identifier["[" const\_int "]"] (\* A normal identifier or an array identifier \*)

factor = identifier | constant

FILE: p1.Z

# Compute the max of 3 numbers

{

# Read the 3 numbers

print("num\_1 = ");

int num\_1 = input(int);

print("num\_2 = ");

int num\_2 = input(int);

print("num\_3 = ");

int num\_3 = input(int);

# Find the maximum and print it

if ((num\_1 > num\_2) and (num\_1 > num\_3))

{

print(num\_1);

}

elif ((num\_2 > num\_1) and (num\_2 > num\_3))

{

print(num\_2);

}

else

{

print(num\_3);

}

}

FILE: p2.Z

# Verify if a number is prime

{

# Read the number

print("nr = ");

int nr = input(int);

int divisor = 2;

int ok = 1;

# Iterate through all numbers in range(2, n/2) and check

# if they divide the number

while (divisor < (nr / 2))

{

if ((nr % divisor) == 0)

{

ok = 0;

}

divisor = divisor + 1;

}

# Print the result

if (ok == 1)

{

print("The number ", nr ," is prime.");

}

else

{

print("The number ", nr ," is not prime.");

}

}

FILE: p3.Z

# Check if a number appears in a given array of numbers

{

# Read the number

int my\_array[30];

int i = 0;

int n = 0;

int sum = 0;

found = 0;

int searched;

print("my\_array[", n, "] = ");

my\_array[n] = input(int);

n = n + 1;

while (my\_array[n-1] != -1)

{

print("my\_array[", n, "] = ");

my\_array[n] = input(int);

n = n + 1;

}

print("searched = ");

searched = input(int);

# Iterate through all numbers in the array and check

# if they are the searched number

while ((i < n) and (found = 0))

{

if (my\_array[i] == searched)

{

print("The number was found at position ", i);

found = 1;

}

}

# If the number wasn't found, print that

if (found == 0)

{

print("The number was not found in the array.");

}

}

FILE: p1err.Z

# Compute the max of 3 numbers

{

# Read the 3 numbers

print("num\_1 = ");

# Error 1: $ is not valid outside a string or comment

$

int num\_1 = input(int);

print("num\_2 = ");

int num\_2 = input(int);

print("num\_3 = ");

# Error 2: 3\_num - variables can only start with latin letters or underscore

int 3\_num = input(int);

# Find the maximum and print it

if ((num\_1 > num\_2) and (num\_1 > num\_3))

{

print(num\_1);

}

elif ((num\_2 > num\_1) and (num\_2 > num\_3))

{

print(num\_2);

}

else

{

print(num\_3);

}

}