a) Minilanguage Specification

1. Language Definition:

1.1 Alphabet:

a. Lowercase letters (a-z) of the English alphabet

b. Underline character ('\_')

c. Decimal digits (0-9)

1.2 Lexic:

a. Special Symbols:

- Operators: +, -, \*, /, :=, <, <=, =, >=, !=

- Separators: (, ), {, }, ,, ;, ., space

b. Reserved Words:

- Data Types: int, char, struct

- Control Flow: if, else, while, do

- Input/Output: read, write

c. Identifiers:

identifier ::= letter { letter | digit }

letter ::= 'a' | 'b' | ... | 'z'

digit ::= '0' | '1' | ... | '9'

d. Constants:

1. Integer Constants:

int\_const ::= [ '+' | '-' ] digit { digit }

2. Character Constants:

char\_const ::= ''' ( letter | digit ) '''

3. String Constants:

string\_const ::= '"' { letter | digit } '"'

2. Syntax Definition (BNF):

a. Syntactical Rules:

program ::= "int" "main" "(" ")" "{" decllist stmtlist "}"

decllist ::= declaration | declaration ";" decllist

declaration ::= IDENTIFIER ":" type ";"

type ::= "int" | "char" | structdecl

structdecl ::= "struct" IDENTIFIER "{" decllist "}"

b. Compound Statement:

stmtlist ::= stmt | stmt ";" stmtlist

stmt ::= simplstmt | structstmt

c. Simple Statement:

simplstmt ::= assignstmt | iostmt

d. Assignment Statement:

assignstmt ::= IDENTIFIER ":=" expression ";"

expression ::= expression addop term | term

term ::= term mulop factor | factor

factor ::= "(" expression ")" | IDENTIFIER | int\_const | char\_const

addop ::= "+" | "-"

mulop ::= "\*" | "/" | "%"

e. Input/Output Statement:

iostmt ::= "read" "(" IDENTIFIER ")" ";" | "write" "(" IDENTIFIER ")" ";"

f. Structured Statement:

structstmt ::= cmpdstmt | ifstmt | whilestmt

cmpdstmt ::= "{" stmtlist "}"

g. Conditional Statement:

ifstmt ::= "if" "(" condition ")" stmt [ "else" stmt ]

condition ::= expression relop expression

relop ::= "<" | "<=" | "=" | "!=" | ">=" | ">"

h. Loop Statement:

whilestmt ::= "while" "(" condition ")" stmt

b) Write a small program into your programming language

int main() {

int num;

int reversed;

int original;

int remainder;

read(num);

original := num;

reversed := 0;

while (num > 0) {

remainder := num % 10;

reversed := reversed \* 10 + remainder;

num := num / 10;

}

if (original = reversed) {

write("Palindrome");

} else {

write("Not Palindrome");

}

}