1.

<program>  **->** “{“ <bloc-decl-var> <bloc-instr> “}”

<bloc-decl-var> **->** <tip-var> <lista-var> | <bloc-decl-var> <tip-var> <lista-var>

<tip-var> **->** “int” | “double” | “float” | <decl-pred>

<decl-pred> **->** “struct” “{“ <bloc-decl-var> “}”

<lista-var> **->** ID, <lista-var>; | ID;

<bloc-instr> **->** <list-instr>

<list-instr> **->** <instr> | <instr> <list-instr>

<instr> **->** <atribuire> | <io> | <cond> | <bloc-decl-var> | <rep>

<atribuire> **->** ID = <expr> “;”

<expr> **->** ID | CONST | ID <op> ID |ID <op> CONST | CONST <op> ID| ID <op> <expr> | <expr> <op> <expr>

<op> **->** “+” | “-“ | “/” | “%” | “\*”

<io> **->** “cin >> ” ID; | “cout << “ <expr> “<< endl” ;

<cond> **->** “if” “(“ <cerinta> “)” “{“ <bloc-instr> “}” | <cond> “else if” “(“ <cerinta> “)” “{“ <bloc-instr> “}”| <cond> “else” “{“ <bloc-inst> “}”

<cerinta> **->** ID <comp> <expr>

<rep> **->** “for” “(“ ID = <expr> “;” ID <comp> CONST “;” ID = <expr> “)” |“for” “(“ ID = <expr> “;” ID   
 <comp> ID “;” ID = <expr> “)” | “while” “(“ <cerinta> “)” “{“ <bloc-instr> “}”

<comp> **->** “<” | “>” | “<=” | “>=” | “==” | “!=”

2.

I.

{

int r;

cin >> r;

cout << 2 \* pi \* r << endl;

cout << pi \* r \* r << endl;

}

II.

{

int a, b, r;

cin >> a;

cin >> b;

while ( b != 0 ) {

r = a % b;

a = b;

b = r; }

cout << a << endl;

}

III.

{

int n, s, x, i;

cin >> n;

s = 0;

for ( i = 1; i <= n; i++) {

cin >> x;

s = s + x; }

cout << s << endl;

}

3.

I.

{

int a, b

cin >> a;

b = a;

cout >> b;

}

II.

{

int a, b, s, v[1000] = {0};

char c;

cin >> c;

cin >> a;

s = 0;

while (a > 0) {

s = s + 1;

a--;

v[a] = s;

}

cout << c << “ “ << v[1];

}

Lab. 2

1.

{

int a;

double b

cin >> a;

cout >> a > 1;

2.

{

int a;

double b, c;

cin >> a;

cout << a;

cin >> b >> c;

}

Rezolvare: bb