#include <iostream>  
  
using namespace std;  
  
void calculate\_perimeter\_and\_area\_of\_circle() {  
 double radius;  
 cin >> radius;  
 double perimeter;

perimeter = 2 \* 3.14 \* radius;  
 double area;

area = 3.14 \* radius \* radius;  
 cout << perimeter;  
 cout << area;  
}  
  
  
int main() {  
 calculate\_perimeter\_and\_area\_of\_circle();  
 return 0;  
}

#include <iostream>  
  
using namespace std;  
  
  
void cmmdc() {  
 int a;

int b;  
 cin >> a >> b;  
 while (a != b) {  
 if (a > b) {  
 a = a - b;  
 }  
 else {  
 b = b - a;  
 }  
 }  
 cout << a;  
}  
  
int main() {  
 cmmdc();  
 return 0;  
}

#include <iostream>  
  
using namespace std;  
  
  
void sum\_of\_numbers() {  
 int n;  
 cin >> n;  
 int sum;

sum = 0;  
 while (n > 0) {  
 int x;  
 cin >> x;  
 sum = sum + x;  
 n = n - 1;  
 }  
 cout << sum;  
}

int main() {  
 sum\_of\_numbers();  
 return 0;  
}

Specificarea minilimbajului

<program>::#include <iostream>

using namespace std;

<declarare\_functii>

<declarare\_functii>::=<functie>< declarare\_functii>|< functie>

<functie>::=<antet><corp>

<antet>::= <tip> ID ()

<tip>::=int | float | struct | double

<lista\_decl>::=<decl> <lista\_decl>|<decl>

<decl>::=<tip><atribuire |<tip>ID;

<corp>::={<instr\_comp>}

<instr\_comp>}::=<instr><instr\_comp>|<instr>

<instr>::=<atribuire>;|<instr\_ciclare>|<instr\_returnare>|<instr\_intrare>|<instr\_iesire>|<decl>|<instr\_if> | <apel\_func>

<instr\_if>::= if (<cond >) <corp> | if (<cond >) <corp> else <corp>

<atribuire>::=ID = <expresie\_aritmetica>

<expresie\_aritmetica>::=<expresie\_aritmetica><op\_aritmetica><expresie\_aritmetica> | ID | CONST

<instr\_intrare>::=cin <lista\_intrari>;

<lista\_intrari>::= >> ID <lista\_intrari> | >> ID

<instr\_ciclare>:: while (<cond>) <corp> | for (<decl> <cond>; <atribuire>) <corp>

<instr\_returnare> ::= return 0;

<apel\_func> ::= ID ();

< cond >::= ID | CONST | <expresie\_aritmetica ><op\_rel>< expresie\_aritmetica>

<instr\_iesire>::= cout <lista\_iesiri>;

<lista\_iesiri>::= << <expresie\_aritmetica> <lista\_iesiri> | << <expresie\_aritmetica>

<op\_rel>::= != | == | < | > | <= | >=

<op\_art>::= + | \* | / | % | -

<instr\_return>::= return <expresie\_aritmetica>;

#include <istream>  
  
using namespace st;  
  
  
// Programul nu compileaza  
void program\_gresit\_1() {

int a;  
 cin >> a;  
 cout << a;  
}  
  
int main() {  
 program\_gresit\_1();  
 return 0;  
}

#include <iostream>  
  
using namespace std;

void program\_gresit\_2() {  
 int a;  
 cin >> a;  
 cout << sqrt(a) << "\n";  
 cout << cbrt(a);  
}