Universitatea “Spiru Haret”

Facultatea de Inginerie si Informatica

Limbaje formale si compilatoare

Subiectul I

Student: Teodorescu Vlad

Grupa: 202

**1.Cerinta**

**I.** Se consideră funcţia bijectivă  dată de unde:

<*x*,*y*> = 2*x*(2*y*+1)-1

şi să se scrie un program care:

* Dacă la intrare apare o pereche de numere naturale x,y produce la ieşire <*x*,*y*>
* Dacă la intrare apare un singur număr natural z, determină numerele naturale x,y cu <x,y>=z

**1.1. Explicarea programului**

**1.1.1. Importarea claselor**

In program vom utiliza doua pachete

* Java.util.Scanner necesara pentru a putea citi de la tastatura numerele/numarul de prelucrat
* Java.lang.Math necesara pentru a utiliza metoda de ridicare la putere

**1.1.2. Metoda readNextValue()**

private static int readNextValue(){  
 int x;  
 /\*  
 while the imput is negative or not a number retry  
 \*/  
 while(true){  
 try {  
 Scanner scaner = new Scanner(System.*in*);  
 x = scaner.nextInt();  
 if (x<0) {  
 /\*  
 generate a new error for negative number  
 \*/  
 throw new RuntimeException();  
 }  
 break;  
 }  
 catch(InputMismatchException e) {  
 System.*out*.println("Nu ati introdus un numar natural! Reincercati");  
 }  
 catch(RuntimeException a) {  
 System.*out*.println("Nu ati introdus un numar natural! Reincercati");  
 }  
 }  
 return x;  
}

Functia intoarce un numar citit de la tastatura verificand daca este integer is daca este pozitiv.

In caz de eroare aplicatia intra in bucla pana cand primeste o valoare valida.

public static void main(String[] args) {  
 Scanner scaner = new Scanner(System.*in*);  
 System.*out*.println("Daca doriti sa introduceti perechea X,Y apasati \"y\" si ENTER");  
 char qbranch='y';  
 /\*  
 daca primul caracter introdus e y cerem de la tastatura x si y  
 \*/  
 if (scaner.next().charAt(0)=='y') {  
 System.*out*.println("Introduceti primul numar!");  
 int x = *readNextValue*();  
 System.*out*.println("Introduceti al doilea numar!");  
 int y = *readNextValue*();  
 int z = ((int) Math.*pow*(2, x) \* ((2 \* y) + 1)) - 1;  
 System.*out*.println(z);  
 }  
 else{  
 // daca primul caracter introdus nu e y cerem de la tastatura rezultatul lui <x,y>  
 System.*out*.println("Introduceti rezultatul lui <x,y>!");  
 //citim rezultatul si adunam 1 (trecut peste egal -1)  
 int z = 1 + *readNextValue*() ;  
 int x = 0; // initializam x cu 0 (x determina de cate ori se imparte exact la 2 z(numarul introdus +1)  
 while (z%2==0){  
 z=z/2;//reducem puterea lui 2 prin impartire la 2  
 x=x+1;// si incrementam x  
 }  
 int y=(z-1)/2;//determinam y in functie de zcare acum este (numarinrodus+1)/2^x  
  
 System.*out*.println("<"+x+","+y+">");  
 }  
}

Intrebam daca asteptam 1 sau 2 elemente

Daca avem 2 elemente avem x is y is calculam rezultatul

int z = ((int) Math.pow(2, x) \* ((2 \* y) + 1)) – 1;

folosind ridicarea la putere cu rezultat intreg (int) Math.pow(2, x)

Daca avem un element avem <x,y> .

la citire o sa salvam z ca <x,y>+1 deoarece:

z = ((int) Math.pow(2, x) \* ((2 \* y) + 1)) – 1

z+1 = (int) Math.pow(2, x) \* ((2 \* y) + 1)

Putem determina x ca numar de impartiri exacte ale lui z+1 la 2 is ramnem cu:

(z+1)/(int) Math.pow(2, x)=(2 \* y) + 1

la final scadem 1 is impartim la 2 ca sa aflam rezultatul.

Universitatea “Spiru Haret”

Facultatea de Inginerie si Informatica

Limbaje formale si compilatoare

Subiectul II

Student: Teodorescu Vlad

Grupa: 202

**2.Cerinta**

**II. Pentru orice n numere naturale a1,...,an se consideră numărul lui Gödel ataşat prin:**



unde *p*1=2, *p*2=3, *p*3=5, ... este şirul numerelor prime.

Să se scrie un program care:

1. Dacă la intrare apar n>1 numere naturale x,y, produce la ieşire numărul lui Gödel ataşat;

**Dacă la intrare apare un singur număr natural z, determină numerele naturale al căror număr Gödel este z.**

**2.1. Explicarea programului**

**2.1.1. Importarea claselor**

In program vom trei doua pachete

* Java.util.Scanner necesara pentru a putea citi de la tastatura numerele/numarul de prelucrat
* Java.lang.Math necesara pentru a utiliza metoda de ridicare la putere
* Java.util.Stack necesara pentru lucrul cu stive

**2.1.2. Metoda readNextValue()**

private static Stack<Integer> readNextValue(){  
 Stack<Integer> x = new Stack<>();  
 while(true) {  
 try {  
 Scanner scaner = new Scanner(System.*in*);  
 String line = scaner.nextLine();  
 String[] arr = line.split(",");  
 for (String s : arr) {  
 if (Integer.*parseInt*(s) < 0) {  
 throw new RuntimeException();  
 } else {  
 x.push(Integer.*parseInt*(s));  
 }  
 }  
 break;  
 } catch (RuntimeException e) {  
 System.*out*.println("Nu ati introdus un numar natural! Reincercati");  
 x.clear();  
 }  
 }  
 return x;  
}

Citim un sit string de la tastatura, delimitat cu ",".

Folosim functia split pentru a desface in vector folosind regular expresion cu ","

Parcurgem vectorul si verificam daca elementele sunt numere si daca sunt pozitive.

Bucla while permite reintroducerea de la tastatura in cazul in care avem caracter sau numar negativ.

Returnam o stiva de numere intregi.

**2.1.3. Metoda grtPrimeNr(int nrExpon)**

private static Stack<Integer> genPrimeNr(int nrExpon) {  
 Stack<Integer> x = new Stack<>();  
 x.push(2);  
 int number = 3 ;  
 while ((nrExpon-1)>0){  
 boolean isprim=true;  
  
  
 for (Integer elm : x) {  
  
 if (number%elm==0){  
 isprim=false;  
 break;  
 }  
 }  
 if (isprim) {  
 x.push(number);  
 nrExpon = nrExpon - 1;  
 }  
 number=number+1;  
 System.*out*.println(x);  
 }  
  
 return x;  
}

Functia creata aici primeste un singur argument numar intreg care reprezinta numarul de exponenti introdusi de la tastatura.

Initializam cu primul numar prim (2) o stiva de numere prime.

Plecam de la 3 si verificam toate numerele daca se divid cu elementele din stiva noastra de numere prime pt a determina urmatorul numar prim.

Decrementam numar de exponenti ramas astfel incat sa generam cate un numar prim pt fiecare exponent.

Returnam stiva de numere prime

**2.1.4. Metoda grtPrimeAfter(int rez)**

private static Stack<Integer> genPrimeAfter(int rez) {  
 Stack<Integer> primeList = new Stack<>();  
 Stack<Integer> expList = new Stack<>();  
  
 primeList.push(2);  
 int number=3;  
 while (number<=rez){  
  
 boolean isprim=true;  
  
  
 for (Integer elm : primeList) {  
  
 if (number%elm==0){  
 isprim=false;  
 break;  
 }  
 }  
 if (isprim) {  
 primeList.push(number);  
 }  
 number=number+1;  
  
 }  
  
 while (rez>1){  
 for (Integer elm : primeList) {  
 expList.push(0);  
 while (rez%elm==0){  
 expList.push(expList.pop()+1);  
 rez=rez/elm;  
 }  
 }  
 }  
 return expList;  
}

Daca am primit un singur element el o sa fie numarul lui Gödel. Functia aceasta primeste numarul lui Gödel ca argument.

Initializam 2 stive, una de numere prime si una de exponenti.

folosim logica similara ca la functia anterioara pt a determina toate numerele prime mai mici decat rezultatul nostru.

Impartim succesiv toate numerele prime pana nu se mai impart exact si contorizam exponentul.

La final intoarcem lista de exponenti.

**2.1.5. Metoda main()**

public static void main(String[] args) {  
 int nrExpon;  
 int rezultat = 1;  
 Stack<Integer> primeList;  
 Stack<Integer> lista;  
  
 System.*out*.println("Introduceti lista de numere delimitata cu \",\" pentru a se calcula numarul lui Gödel. Pentru calcularea listei de numere introduceti doar un singur element!");  
 lista=*readNextValue*();  
 nrExpon=lista.size();  
 if (nrExpon==1) {  
 int rez= lista.pop();  
 lista=*genPrimeAfter*(rez);  
 System.*out*.println("Lista exponentilor este:");  
 System.*out*.println(lista);  
 }  
 else{  
 primeList = *genPrimeNr*(nrExpon);  
  
 while (!lista.isEmpty()) {  
 rezultat = rezultat \* ((int) Math.*pow*(primeList.pop(), lista.pop()));  
 }  
 System.*out*.println("Numarul lui Gödel este:");  
 System.*out*.println(rezultat);  
 }  
  
  
}

Citim de la tastatura, daca avem un singur element in stiva apelam functia care determina exponentii grtPrimeAfter(int rez), altfel apelam functia care determina stiva de numere prime la care aplicam functia ca sa determinam numarul lui Gödel .

Numărul lui Gödel este calculat aici folosind functia pop() care returneaza ultimul element al stivei si il elimina din stiva. astfel incat eliminam un numar prim si un exponent pe rand calculam nrPrim^exponent si inmultim cu rezultatul final (initializat cu 1 ca element neutru al inmultirii)