Q1)

Write a java program to check whether a number is emirp number or not

Example:

13, 17, 31, 37, 71, 73, 79, 97, 107, 113, 149, 157, 167, 179, 199

Algorithm

1. START

2. take input form user

3. execute a loop from i=2 to i<(n/2)

4. check if n is divisible by i from 2 to n/2

5. if i is not divisible by any value of i then n is prime

6. if i is divisible by any value of i then print "invalid input" and exit program

7. to find the reverse of n calculate the n%10 and add to the reverse number

8. update n by n/10

9. Now check if the reverse number is prime or not

10. Repeat steps 3 to 5 to check for prime number

11. if reverse number is prime number print n is an Emirp number otherwise print n is not an Emirp number

12. END

Source Code

import java.util.Scanner;

public class Emirp{

public static boolean isPrime(int num){

if(num<=1)

return false;

for(int i=2;i<(num/2);i++){

if(num%i==0)

return false;

}

return true;

}

public static void main(String args[]){

Scanner nrt=new Scanner(System.in);

System.out.println("enter a number");

int user=nrt.nextInt();

if(isPrime(user)){

//reversing the number

int copy=user,reverse=0;

while(copy!=0){

reverse = (reverse\*10) +(copy%10);

copy/=10;

}

if(isPrime(reverse))

System.out.println(user+" is an Emirp Number");

else

System.out.println(user+" is NOT an Emirp Number");

}

else

System.out.println("Invalid Input");

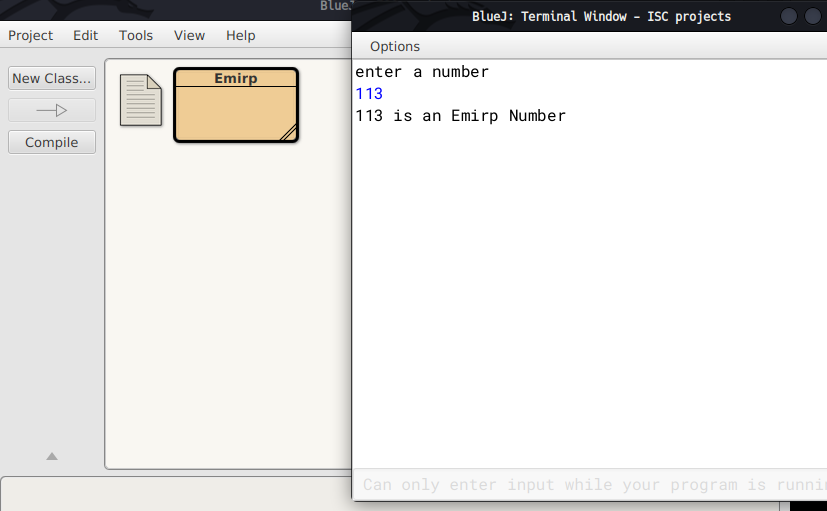
}

}

Variable Description Table

|  |  |  |
| --- | --- | --- |
| Variable name | Data type | Use |
| num | int | Argument for isPrime method |
| nrt | Wrapper object | For taking input |
| user | int | To store the input |
| copy | int | To store a copy of user variable’s value |
| reverse | int | To store the reverse of user |

Output



Q2)

A Circular Prime is a prime number that remains prime under cyclic shifts of its digits. When the leftmost digit is removed and replaced at the end of the remaining string of digits, the generated number is still prime. The process is repeated until the original number is reached again.

A number is said to be prime if it has only two factors 1 and itself.

Example:

131

311

113

Hence, 131 is a circular prime.

Write a program to accept a positive number N and check whether it is a circular prime or not. The new numbers formed after the shifting of the digits should also be displayed.

Test your program with the following data and some random data:

Example 1

INPUT:

N = 197

OUTPUT:

197

971

719

197 IS A CIRCULAR PRIME.

Example 2

INPUT:

N = 1193

OUTPUT:

1193

1931

9311

3119

1193 IS A CIRCULAR PRIME.

Example 3

INPUT:

N = 29

OUTPUT:

29

92

29 IS NOT A CIRCULAR PRIME.

Algorithm

1. Start.

2. Input a number n.

3. Find if the number is prime.

4. Declare a function isPrime() to check prime.

5. Execute loop from i=1 to i<=n.

6. If n is divisible by any i then count the number of times it gets divisible by doing c++.

7. If c==2, then the number n is prime .

8. If not then stop otherwise go to next step.

9. Calculate the length (l) of the number by converting n into string.

10. Then find the divisor=(int)(Math.pow(10,l-1)).

11. Store a copy of n in m.

12. Execute a loop from i=0 to i<l to generate new circulated numbers.

13. Calculate the quotient=n1/divisor and the remainder=n2%divisor.

14. To generate the new circulated number(m) follow the next step.

15. m=r\*10+n1;

16. Call the function isPrime() to check whether new number(m) is prime.

17. If m is not prime then break;

18. If all the numbers are prime then print it is a circular prime otherwise not a circular

prime.

19. Stop

Source Code

import java.util.Scanner;

class CircularPrime{

static boolean isPrime(int num) {

int c = 0;

for (int i = 1; i <= num; i++)

{ if (num % i == 0){

c++;

}

}

if(c==2)

System.out.println(num+"is prime");

else

System.out.println(num+"is not prime");

return c == 2;

}

public static void main(String args[]){

Scanner in = new Scanner(System.in);

System.out.print("Enter the number: ");

int n = in.nextInt();

int f=1;

if (isPrime(n)){

//System.out.println(n);

String s=Integer.toString(n);

int l=s.length();

int divisor = (int)(Math.pow(10, l- 1));

int m = n;

for (int i = 1; i < l; i++){

int n1 = m / divisor;

int n2 = m % divisor;

m = n2 \* 10 + n1;

//System.out.println(m);

if (!isPrime(m)) {

f=0;

break;

}

}

}

else {

f=0;

}

if (f==1){

System.out.println(n + " is a circular prime.");

}

else {

System.out.println(n + " is not a circular prime.");

}

}

}

Variable Description Table

|  |  |  |
| --- | --- | --- |
| Variable name | Data type | Use |
| num | int | Argument for isPrime method |
| in | Wrapper object | For taking input |
| n | int | To store the input |
| m | int | Stores a copy of n |
| n1 | int | Stores quotient |
| n2 | int | divisor |
| copy | int | To store a copy of user variable’s value |
| c | int | To check if a number is prime or not |
| l | int | Stores length of n |
| s | String | Stores String form of n |
| divisor | int | Stores the divisor |

Output

