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”);

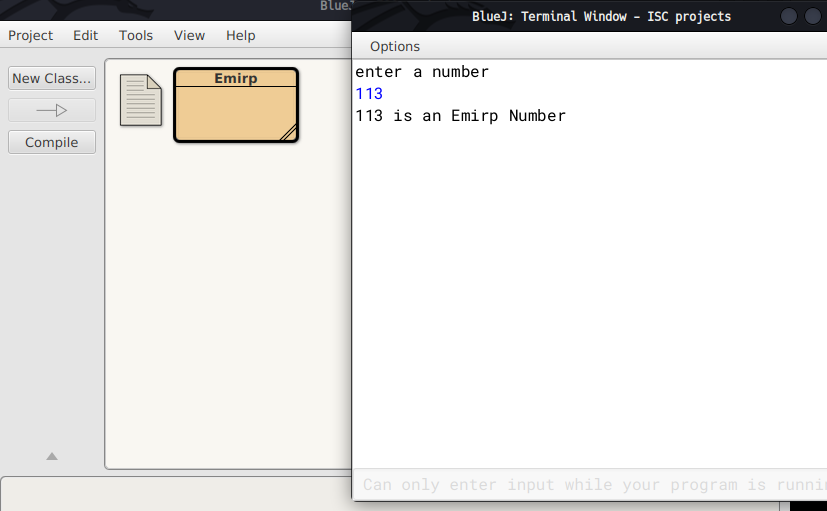
}

}

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



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.");

}

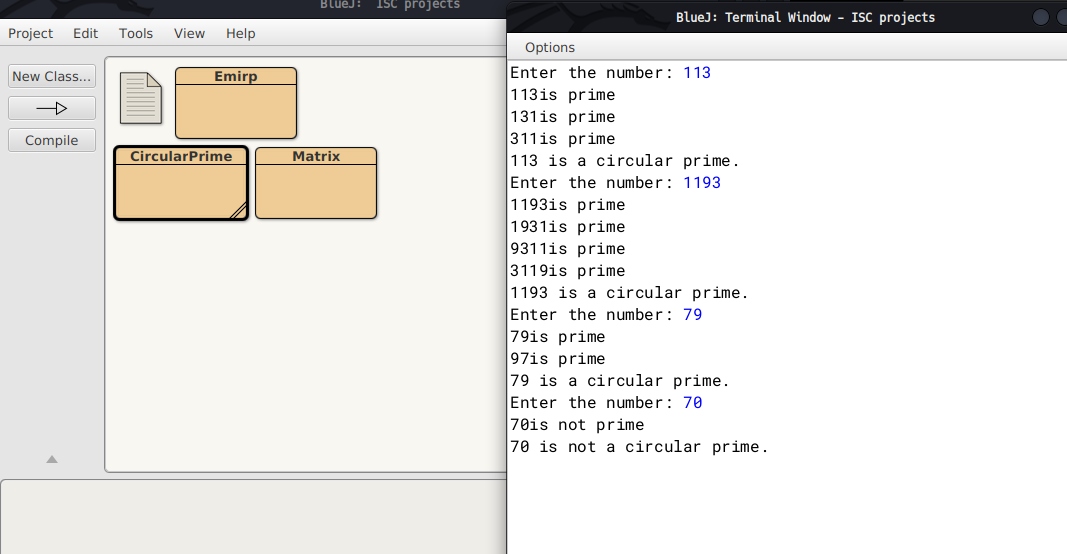
}

}

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



Q3)

3. Write a program to declare a square matrix M [ ] [ ] of order ‘N’ where ‘N’ must be greater than 3 and less than 10. Allow the user to accept three different characters from the keyboard and fill the array according to the instruction given below:

(i) Fill the four corners of the square matrix by character 1.

(ii) Fill the boundary elements of the matrix (except the four corners) by character 2.

(iii) Fill the non-boundary elements of the matrix by character 3.

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

INPUT: N = 5

FIRST CHARACTER: A

SECOND CHARACTER: C

THIRD CHARACTER: X

OUTPUT:

A C C C A

C X X X C

C X X X C

C X X X C

A C C C A

INPUT: N = 4

FIRST CHARACTER: @

SECOND CHARACTER: ?

THIRD CHARACTER: #

OUTPUT:

@ ? ? @

? # # ?

? # # ?

@ ? ? @

Algorithm

1 START

2 Ask user to input three characters.

3 Ask user to input the order of matrix 'M'.

4 Declare a two dimensional array of type char with name 'M'.

5 Start a for loop for(int i=0;i<n;i++).

6 Start a for loop inside as for(int j=0;j<n;j++)

7 If (i and j are both zero) or (either i is zero and j is n-1) or (i is

n-1 and j is zero) or (both i and j are n-1), then

fill two dimensional array M at location M[i][j] with the

first character the user has given as input.

8 If i and j lies between 1 and n-2 then fill two dimensional array M at

location M[i][j] with the second character user has given.

9 Else fill M[i][j] with the third character given as input by the user.

10 Run two nested loops as for(int i=0;i<n;i++) and inside

for(int j=0;j<n;j++) and print all elements.

11 END

Source Code

import java.util.Scanner;

public class Matrix{

public static void main(){

Scanner nrt=new Scanner(System.in);

System.out.println("Enter three characters");

char ch1,ch2,ch3;

ch1=(nrt.nextLine()).charAt(0);

ch2=(nrt.nextLine()).charAt(0);

ch3=(nrt.nextLine()).charAt(0);

System.out.println("Enter order of Matrix");

int n=nrt.nextInt();

char M[][]= new char[n][n];

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

if( (i==0 && j==0) || (i==0 && j==n-1) || (i==n-1 && j==0) || (i==n-1 && j==n-1))

M[i][j]=ch1;

else if ( (i>=1 && i<=(n-2)) && (j>=1 && j<=(n-2)))

M[i][j]=ch2;

else

M[i][j]=ch3;

}

}

//printing the matrix

for(int i=0 ;i<n;i++){

for(int j=0; j<n;j++){

System.out.print("\t"+M[i][j]);

}

System.out.println();

}

nrt.close();

}

}

Variable Description Table

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| char | M | Two dimensional array to store the characters |
| int | i ,j | Loop variables |
| Int | n | Stores the order of the matrix |
| char | ch1 ,ch2,ch3 | Stores the first, second and third characters given as input by the user respectively |

Output

