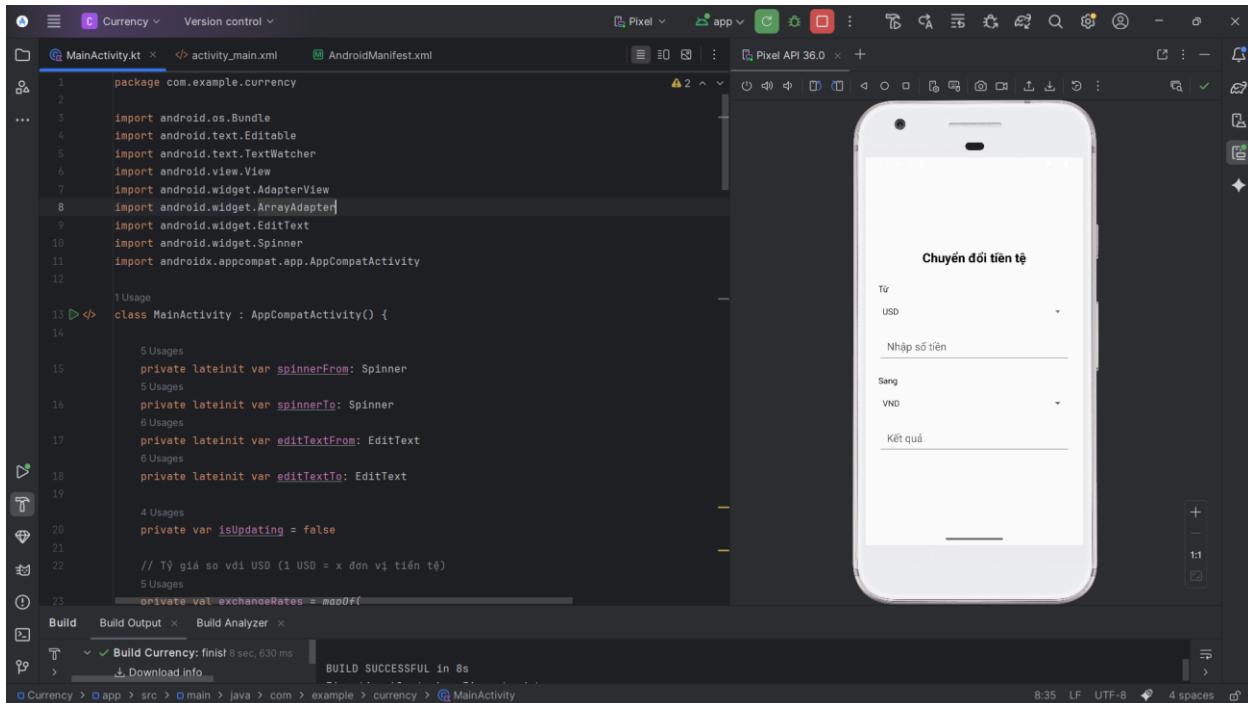


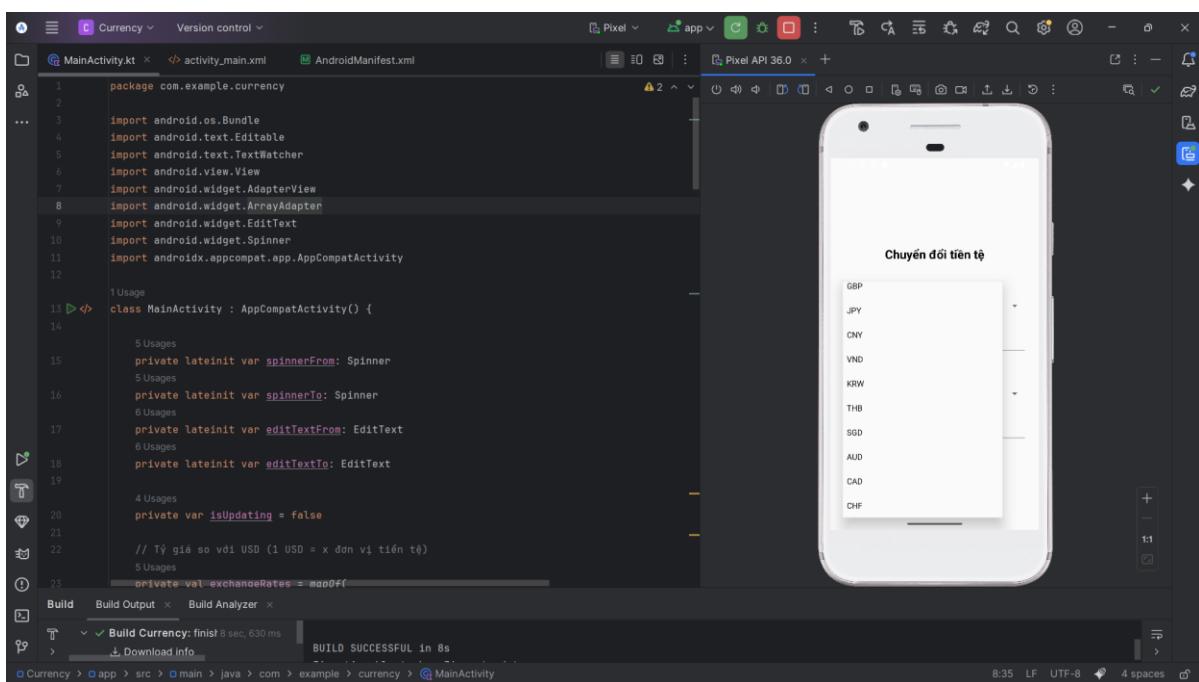
# Bài tập Android Week8 - 4/11/2025

Bài 1: Chuyển đổi tiền tệ [https://github.com/NKDuyennn/IT4785\\_android\\_ex1\\_week8](https://github.com/NKDuyennn/IT4785_android_ex1_week8)

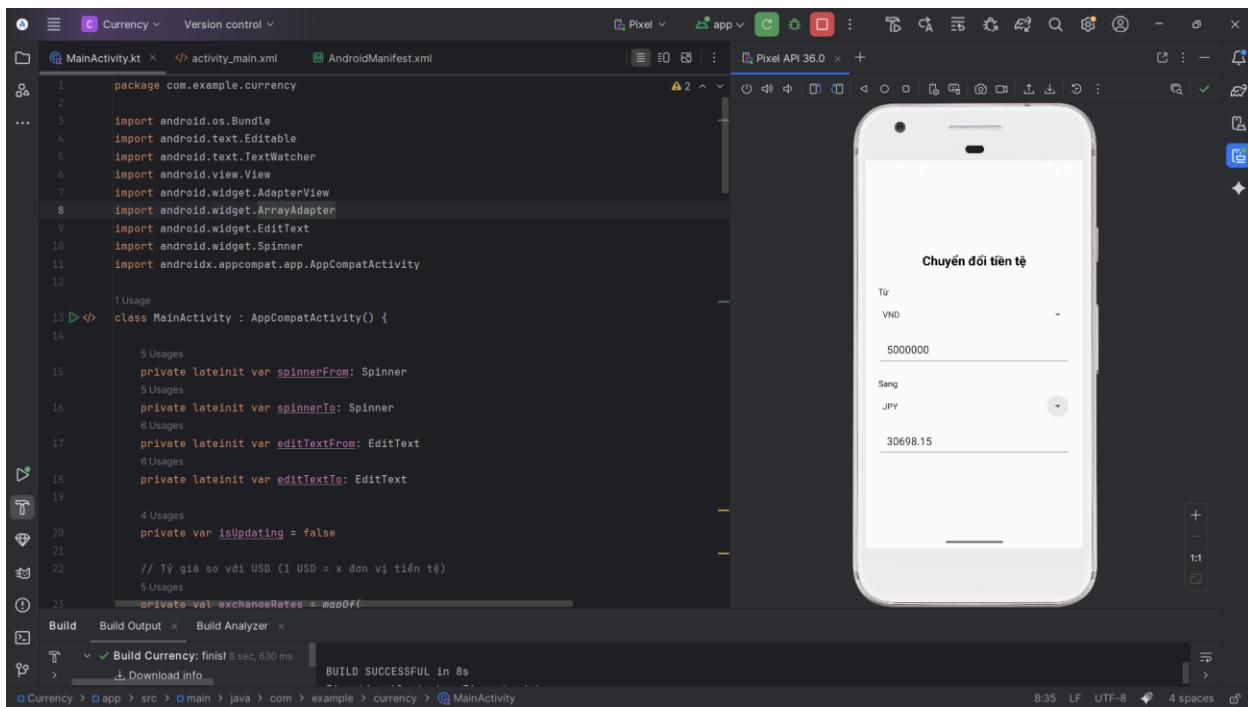
Chạy chương trình



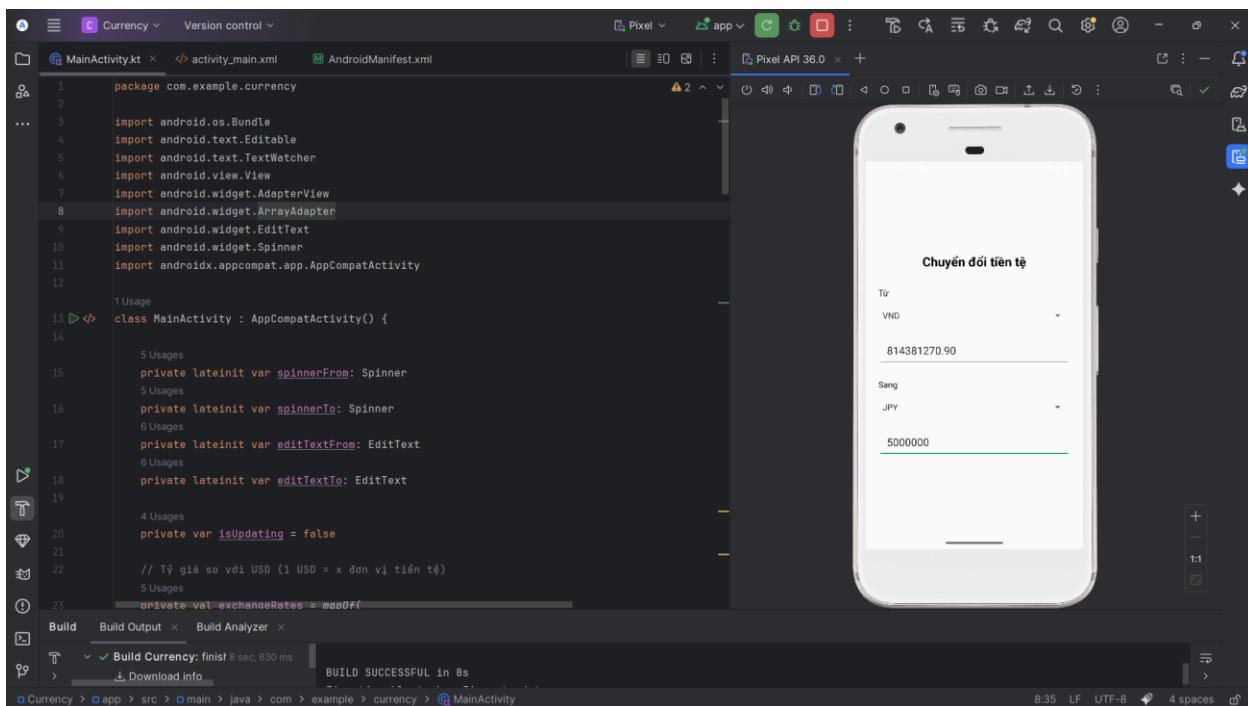
Chọn loại tiền tệ muốn chuyển



## Chuyển từ VND -> JPY



## Chuyển từ JPY -> VND



## Bài 2: Number List [https://github.com/NKDuyennn/IT4785\\_android\\_ex2\\_week8](https://github.com/NKDuyennn/IT4785_android_ex2_week8)

Chạy chương trình

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "NumberList". The `MainActivity.kt` file is open in the editor.
- Code Editor:** The `MainActivity.kt` file contains Java code for checking if a number is perfect, square, or Fibonacci. The code includes comments and three separate functions: `isPerfect`, `isSquare`, and `isFibonacci`.
- Emulator:** A Pixel API 36.0 emulator is running, displaying a list of numbers from 2 to 22. The first few items in the list are:
  - 2
  - 4
  - 6
  - 8
  - 10
  - 12
  - 14
  - 16
  - 18
  - 20
  - 22
- Toolbar:** The toolbar at the top includes icons for Pixel, app, and various developer tools like Lint, Build, and Run.
- Status Bar:** The bottom status bar shows the device is connected via USB (193.2), using LF encoding, and has 4 spaces.

Số chẵn

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The project is named "NumberList". The `MainActivity.kt` file is open in the editor.
- Code Editor:** The `MainActivity.kt` file contains Java code for checking if a number is perfect, square, or Fibonacci. The code includes comments and three separate functions: `isPerfect`, `isSquare`, and `isFibonacci`.
- Emulator:** A Pixel API 36.0 emulator is running, displaying a list of even numbers from 2 to 22. The list is:
  - 2
  - 4
  - 6
  - 8
  - 10
  - 12
  - 14
  - 16
  - 18
  - 20
  - 22
- Toolbar:** The toolbar at the top includes icons for Pixel, app, and various developer tools like Lint, Build, and Run.
- Status Bar:** The bottom status bar shows the device is connected via USB (193.2), using LF encoding, and has 4 spaces.

## Số lẻ

The screenshot shows the Android Studio interface with the project 'NumberList' open. The code editor displays the MainActivity.kt file, which contains Java code for finding perfect numbers, checking if a number is a square, and generating Fibonacci numbers. The run tab shows an iPhone X emulator with a list of odd numbers from 1 to 21. The 'Số lẻ' checkbox is checked, while others like 'Số chẵn', 'Số nguyên tố', 'Số hoàn hảo', 'Số chính phương', and 'Số Fibonacci' are unchecked.

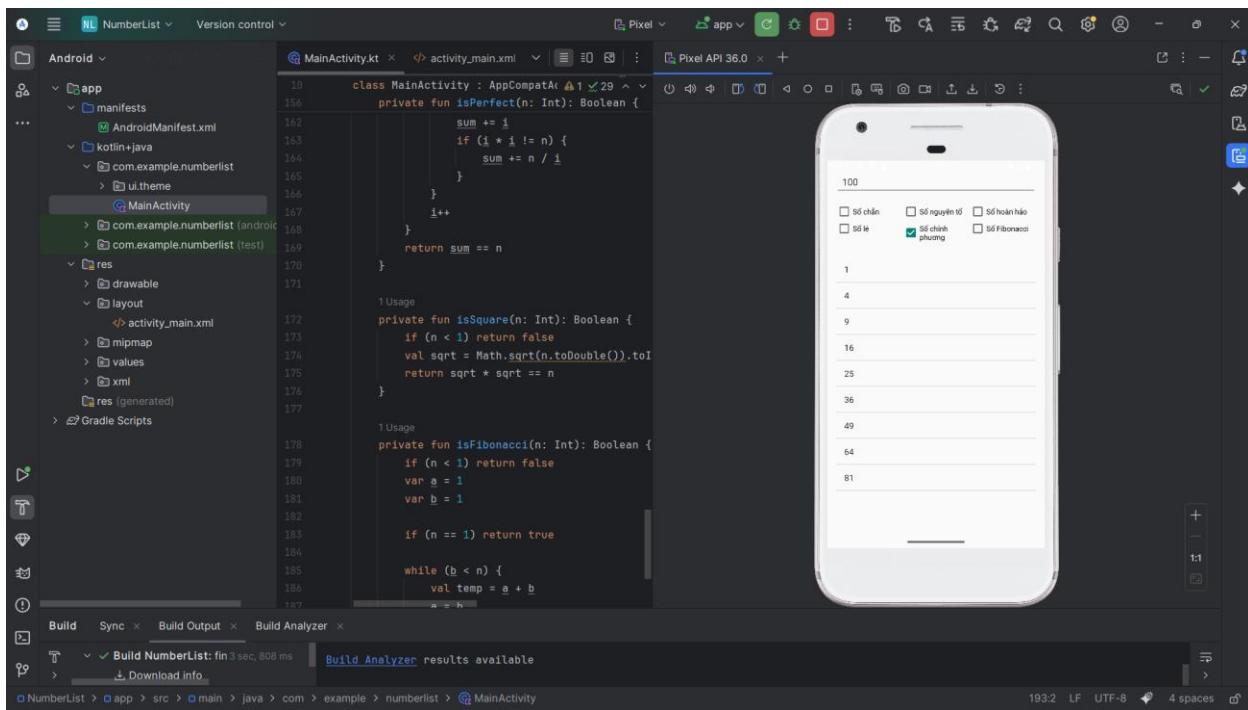
```
10    class MainActivity : AppCompatActivity() {
11        private fun isPerfect(n: Int): Boolean {
12            var sum = 1
13            for (i in 2..n/2) {
14                if (i * i == n) {
15                    sum += i
16                }
17            }
18            return sum == n
19        }
20
21        private fun isSquare(n: Int): Boolean {
22            if (n < 1) return false
23            val sqrt = Math.sqrt(n.toDouble()).toI
24            return sqrt * sqrt == n
25        }
26
27        private fun isFibonacci(n: Int): Boolean {
28            if (n < 1) return false
29            var a = 1
30            var b = 1
31
32            if (n == 1) return true
33
34            while (b < n) {
35                val temp = a + b
36                a = b
37                b = temp
38            }
39            return b == n
40        }
41    }
```

## Số nguyên tố

The screenshot shows the Android Studio interface with the project 'NumberList' open. The code editor displays the MainActivity.kt file, which contains Java code for finding perfect numbers, checking if a number is a square, and generating Fibonacci numbers. The run tab shows an iPhone X emulator with a list of prime numbers from 2 to 31. The 'Số nguyên tố' checkbox is checked, while others like 'Số chẵn', 'Số lẻ', 'Số hoàn hảo', 'Số chính phương', and 'Số Fibonacci' are unchecked.

```
10    class MainActivity : AppCompatActivity() {
11        private fun isPerfect(n: Int): Boolean {
12            var sum = 1
13            for (i in 2..n/2) {
14                if (i * i == n) {
15                    sum += i
16                }
17            }
18            return sum == n
19        }
20
21        private fun isSquare(n: Int): Boolean {
22            if (n < 1) return false
23            val sqrt = Math.sqrt(n.toDouble()).toI
24            return sqrt * sqrt == n
25        }
26
27        private fun isFibonacci(n: Int): Boolean {
28            if (n < 1) return false
29            var a = 1
30            var b = 1
31
32            if (n == 1) return true
33
34            while (b < n) {
35                val temp = a + b
36                a = b
37                b = temp
38            }
39            return b == n
40        }
41    }
```

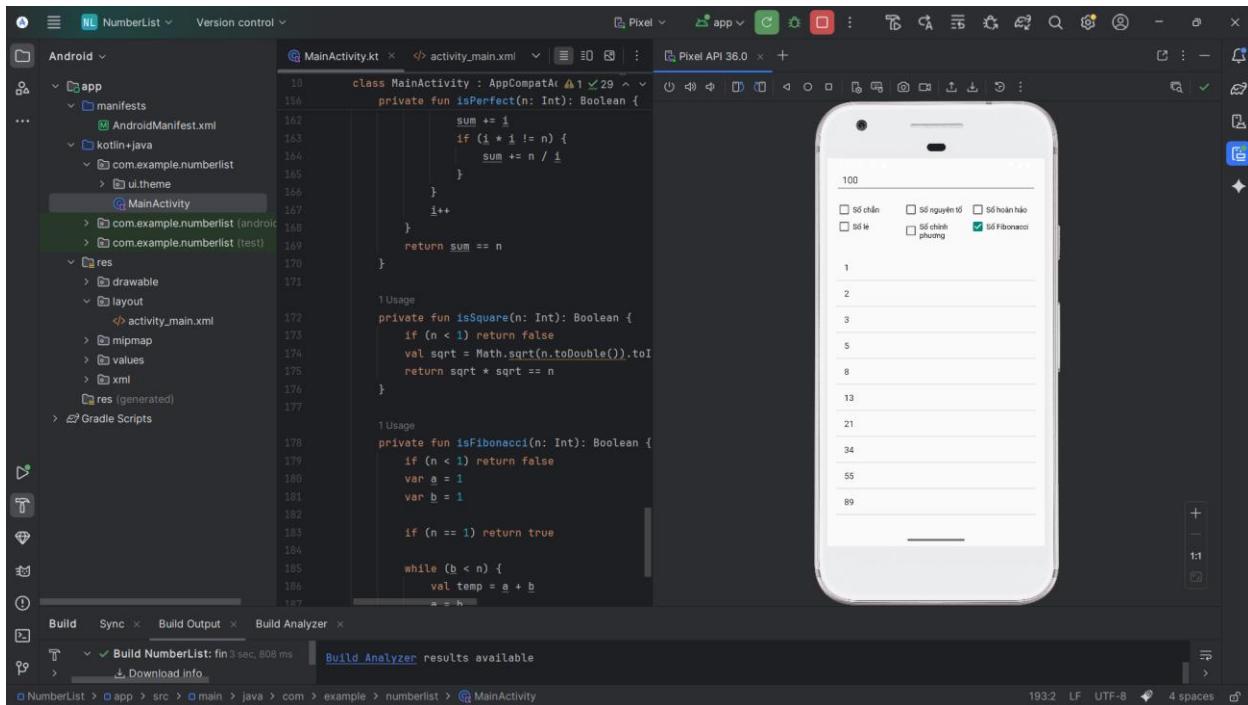
## Số chính phương



The screenshot shows the Android Studio interface with the project 'NumberList' open. The code editor displays the MainActivity.kt file, which contains logic to find perfect squares. The emulator window shows a list of numbers from 1 to 100, with checkboxes for various number types. The 'Số chính phương' checkbox is checked, and the list shows 1, 4, 9, 16, 25, 36, 49, 64, and 81.

```
10     class MainActivity : AppCompatActivity() {
11         private fun isPerfect(n: Int): Boolean {
12             var sum = 0
13             for (i in 1..n) {
14                 if (i * i == n) {
15                     sum += i
16                 }
17             }
18             return sum == n
19         }
20
21         private fun isSquare(n: Int): Boolean {
22             if (n < 1) return false
23             val sqrt = Math.sqrt(n.toDouble()).toI
24             return sqrt * sqrt == n
25         }
26
27         private fun isFibonacci(n: Int): Boolean {
28             if (n < 1) return false
29             var a = 1
30             var b = 1
31
32             while (b < n) {
33                 val temp = a + b
34                 a = b
35                 b = temp
36             }
37             return b == n
38         }
39
40         private fun isPrime(n: Int): Boolean {
41             if (n < 2) return false
42             for (i in 2..n - 1) {
43                 if (n % i == 0) return false
44             }
45             return true
46         }
47
48         private fun isDivisibleBy5(n: Int): Boolean {
49             return n % 5 == 0
50         }
51
52         private fun isDivisibleBy3(n: Int): Boolean {
53             return n % 3 == 0
54         }
55
56         private fun isDivisibleBy7(n: Int): Boolean {
57             return n % 7 == 0
58         }
59
60         private fun isDivisibleBy11(n: Int): Boolean {
61             return n % 11 == 0
62         }
63
64         private fun isDivisibleBy13(n: Int): Boolean {
65             return n % 13 == 0
66         }
67
68         private fun isDivisibleBy17(n: Int): Boolean {
69             return n % 17 == 0
70         }
71
72         private fun isDivisibleBy19(n: Int): Boolean {
73             return n % 19 == 0
74         }
75
76         private fun isDivisibleBy23(n: Int): Boolean {
77             return n % 23 == 0
78         }
79
80         private fun isDivisibleBy29(n: Int): Boolean {
81             return n % 29 == 0
82         }
83
84         private fun isDivisibleBy31(n: Int): Boolean {
85             return n % 31 == 0
86         }
87
88         private fun isDivisibleBy37(n: Int): Boolean {
89             return n % 37 == 0
90         }
91
92         private fun isDivisibleBy41(n: Int): Boolean {
93             return n % 41 == 0
94         }
95
96         private fun isDivisibleBy43(n: Int): Boolean {
97             return n % 43 == 0
98         }
99
100        private fun isDivisibleBy47(n: Int): Boolean {
101            return n % 47 == 0
102        }
103
104        private fun isDivisibleBy53(n: Int): Boolean {
105            return n % 53 == 0
106        }
107
108        private fun isDivisibleBy59(n: Int): Boolean {
109            return n % 59 == 0
110        }
111
112        private fun isDivisibleBy61(n: Int): Boolean {
113            return n % 61 == 0
114        }
115
116        private fun isDivisibleBy67(n: Int): Boolean {
117            return n % 67 == 0
118        }
119
120        private fun isDivisibleBy71(n: Int): Boolean {
121            return n % 71 == 0
122        }
123
124        private fun isDivisibleBy73(n: Int): Boolean {
125            return n % 73 == 0
126        }
127
128        private fun isDivisibleBy79(n: Int): Boolean {
129            return n % 79 == 0
130        }
131
132        private fun isDivisibleBy83(n: Int): Boolean {
133            return n % 83 == 0
134        }
135
136        private fun isDivisibleBy89(n: Int): Boolean {
137            return n % 89 == 0
138        }
139
140        private fun isDivisibleBy97(n: Int): Boolean {
141            return n % 97 == 0
142        }
143
144        private fun isDivisibleBy101(n: Int): Boolean {
145            return n % 101 == 0
146        }
147
148        private fun isDivisibleBy103(n: Int): Boolean {
149            return n % 103 == 0
150        }
151
152        private fun isDivisibleBy107(n: Int): Boolean {
153            return n % 107 == 0
154        }
155
156        private fun isDivisibleBy109(n: Int): Boolean {
157            return n % 109 == 0
158        }
159
160        private fun isDivisibleBy113(n: Int): Boolean {
161            return n % 113 == 0
162        }
163
164        private fun isDivisibleBy127(n: Int): Boolean {
165            return n % 127 == 0
166        }
167
168        private fun isDivisibleBy131(n: Int): Boolean {
169            return n % 131 == 0
170        }
171
172        private fun isDivisibleBy137(n: Int): Boolean {
173            return n % 137 == 0
174        }
175
176        private fun isDivisibleBy139(n: Int): Boolean {
177            return n % 139 == 0
178        }
179
180        private fun isDivisibleBy149(n: Int): Boolean {
181            return n % 149 == 0
182        }
183
184        private fun isDivisibleBy151(n: Int): Boolean {
185            return n % 151 == 0
186        }
187
188        private fun isDivisibleBy157(n: Int): Boolean {
189            return n % 157 == 0
190        }
191
192        private fun isDivisibleBy163(n: Int): Boolean {
193            return n % 163 == 0
194        }
195
196        private fun isDivisibleBy167(n: Int): Boolean {
197            return n % 167 == 0
198        }
199
200        private fun isDivisibleBy173(n: Int): Boolean {
201            return n % 173 == 0
202        }
203
204        private fun isDivisibleBy179(n: Int): Boolean {
205            return n % 179 == 0
206        }
207
208        private fun isDivisibleBy181(n: Int): Boolean {
209            return n % 181 == 0
210        }
211
212        private fun isDivisibleBy191(n: Int): Boolean {
213            return n % 191 == 0
214        }
215
216        private fun isDivisibleBy193(n: Int): Boolean {
217            return n % 193 == 0
218        }
219
220        private fun isDivisibleBy197(n: Int): Boolean {
221            return n % 197 == 0
222        }
223
224        private fun isDivisibleBy199(n: Int): Boolean {
225            return n % 199 == 0
226        }
227
228        private fun isDivisibleBy211(n: Int): Boolean {
229            return n % 211 == 0
230        }
231
232        private fun isDivisibleBy223(n: Int): Boolean {
233            return n % 223 == 0
234        }
235
236        private fun isDivisibleBy227(n: Int): Boolean {
237            return n % 227 == 0
238        }
239
240        private fun isDivisibleBy233(n: Int): Boolean {
241            return n % 233 == 0
242        }
243
244        private fun isDivisibleBy239(n: Int): Boolean {
245            return n % 239 == 0
246        }
247
248        private fun isDivisibleBy251(n: Int): Boolean {
249            return n % 251 == 0
250        }
251
252        private fun isDivisibleBy257(n: Int): Boolean {
253            return n % 257 == 0
254        }
255
256        private fun isDivisibleBy263(n: Int): Boolean {
257            return n % 263 == 0
258        }
259
260        private fun isDivisibleBy269(n: Int): Boolean {
261            return n % 269 == 0
262        }
263
264        private fun isDivisibleBy271(n: Int): Boolean {
265            return n % 271 == 0
266        }
267
268        private fun isDivisibleBy283(n: Int): Boolean {
269            return n % 283 == 0
270        }
271
272        private fun isDivisibleBy293(n: Int): Boolean {
273            return n % 293 == 0
274        }
275
276        private fun isDivisibleBy311(n: Int): Boolean {
277            return n % 311 == 0
278        }
279
280        private fun isDivisibleBy331(n: Int): Boolean {
281            return n % 331 == 0
282        }
283
284        private fun isDivisibleBy347(n: Int): Boolean {
285            return n % 347 == 0
286        }
287
288        private fun isDivisibleBy359(n: Int): Boolean {
289            return n % 359 == 0
290        }
291
292        private fun isDivisibleBy367(n: Int): Boolean {
293            return n % 367 == 0
294        }
295
296        private fun isDivisibleBy373(n: Int): Boolean {
297            return n % 373 == 0
298        }
299
299        private fun isDivisibleBy389(n: Int): Boolean {
300            return n % 389 == 0
301        }
302
303        private fun isDivisibleBy401(n: Int): Boolean {
304            return n % 401 == 0
305        }
306
307        private fun isDivisibleBy421(n: Int): Boolean {
308            return n % 421 == 0
309        }
310
311        private fun isDivisibleBy433(n: Int): Boolean {
312            return n % 433 == 0
313        }
314
315        private fun isDivisibleBy463(n: Int): Boolean {
316            return n % 463 == 0
317        }
318
319        private fun isDivisibleBy479(n: Int): Boolean {
320            return n % 479 == 0
321        }
322
323        private fun isDivisibleBy487(n: Int): Boolean {
324            return n % 487 == 0
325        }
326
327        private fun isDivisibleBy503(n: Int): Boolean {
328            return n % 503 == 0
329        }
330
331        private fun isDivisibleBy523(n: Int): Boolean {
332            return n % 523 == 0
333        }
334
335        private fun isDivisibleBy541(n: Int): Boolean {
336            return n % 541 == 0
337        }
338
339        private fun isDivisibleBy563(n: Int): Boolean {
340            return n % 563 == 0
341        }
342
343        private fun isDivisibleBy587(n: Int): Boolean {
344            return n % 587 == 0
345        }
346
347        private fun isDivisibleBy601(n: Int): Boolean {
348            return n % 601 == 0
349        }
350
351        private fun isDivisibleBy613(n: Int): Boolean {
352            return n % 613 == 0
353        }
354
355        private fun isDivisibleBy653(n: Int): Boolean {
356            return n % 653 == 0
357        }
358
359        private fun isDivisibleBy677(n: Int): Boolean {
360            return n % 677 == 0
361        }
362
363        private fun isDivisibleBy701(n: Int): Boolean {
364            return n % 701 == 0
365        }
366
367        private fun isDivisibleBy727(n: Int): Boolean {
368            return n % 727 == 0
369        }
370
371        private fun isDivisibleBy751(n: Int): Boolean {
372            return n % 751 == 0
373        }
374
375        private fun isDivisibleBy787(n: Int): Boolean {
376            return n % 787 == 0
377        }
378
379        private fun isDivisibleBy809(n: Int): Boolean {
380            return n % 809 == 0
381        }
382
383        private fun isDivisibleBy839(n: Int): Boolean {
384            return n % 839 == 0
385        }
386
387        private fun isDivisibleBy863(n: Int): Boolean {
388            return n % 863 == 0
389        }
390
391        private fun isDivisibleBy907(n: Int): Boolean {
392            return n % 907 == 0
393        }
394
395        private fun isDivisibleBy937(n: Int): Boolean {
396            return n % 937 == 0
397        }
398
399        private fun isDivisibleBy967(n: Int): Boolean {
400            return n % 967 == 0
401        }
402
403        private fun isDivisibleBy1009(n: Int): Boolean {
404            return n % 1009 == 0
405        }
406
407        private fun isDivisibleBy1031(n: Int): Boolean {
408            return n % 1031 == 0
409        }
410
411        private fun isDivisibleBy1063(n: Int): Boolean {
412            return n % 1063 == 0
413        }
414
415        private fun isDivisibleBy1091(n: Int): Boolean {
416            return n % 1091 == 0
417        }
418
419        private fun isDivisibleBy1123(n: Int): Boolean {
420            return n % 1123 == 0
421        }
422
423        private fun isDivisibleBy1151(n: Int): Boolean {
424            return n % 1151 == 0
425        }
426
427        private fun isDivisibleBy1187(n: Int): Boolean {
428            return n % 1187 == 0
429        }
430
431        private fun isDivisibleBy1223(n: Int): Boolean {
432            return n % 1223 == 0
433        }
434
435        private fun isDivisibleBy1259(n: Int): Boolean {
436            return n % 1259 == 0
437        }
438
439        private fun isDivisibleBy1283(n: Int): Boolean {
440            return n % 1283 == 0
441        }
442
443        private fun isDivisibleBy1301(n: Int): Boolean {
444            return n % 1301 == 0
445        }
446
447        private fun isDivisibleBy1349(n: Int): Boolean {
448            return n % 1349 == 0
449        }
450
451        private fun isDivisibleBy1367(n: Int): Boolean {
452            return n % 1367 == 0
453        }
454
455        private fun isDivisibleBy1381(n: Int): Boolean {
456            return n % 1381 == 0
457        }
458
459        private fun isDivisibleBy1423(n: Int): Boolean {
460            return n % 1423 == 0
461        }
462
463        private fun isDivisibleBy1451(n: Int): Boolean {
464            return n % 1451 == 0
465        }
466
467        private fun isDivisibleBy1483(n: Int): Boolean {
468            return n % 1483 == 0
469        }
470
471        private fun isDivisibleBy1511(n: Int): Boolean {
472            return n % 1511 == 0
473        }
474
475        private fun isDivisibleBy1549(n: Int): Boolean {
476            return n % 1549 == 0
477        }
478
479        private fun isDivisibleBy1571(n: Int): Boolean {
480            return n % 1571 == 0
481        }
482
483        private fun isDivisibleBy1607(n: Int): Boolean {
484            return n % 1607 == 0
485        }
486
487        private fun isDivisibleBy1643(n: Int): Boolean {
488            return n % 1643 == 0
489        }
490
491        private fun isDivisibleBy1673(n: Int): Boolean {
492            return n % 1673 == 0
493        }
494
495        private fun isDivisibleBy1703(n: Int): Boolean {
496            return n % 1703 == 0
497        }
498
499        private fun isDivisibleBy1733(n: Int): Boolean {
500            return n % 1733 == 0
501        }
502
503        private fun isDivisibleBy1763(n: Int): Boolean {
504            return n % 1763 == 0
505        }
506
507        private fun isDivisibleBy1791(n: Int): Boolean {
508            return n % 1791 == 0
509        }
510
511        private fun isDivisibleBy1823(n: Int): Boolean {
512            return n % 1823 == 0
513        }
514
515        private fun isDivisibleBy1853(n: Int): Boolean {
516            return n % 1853 == 0
517        }
518
519        private fun isDivisibleBy1889(n: Int): Boolean {
520            return n % 1889 == 0
521        }
522
523        private fun isDivisibleBy1919(n: Int): Boolean {
524            return n % 1919 == 0
525        }
526
527        private fun isDivisibleBy1951(n: Int): Boolean {
528            return n % 1951 == 0
529        }
530
531        private fun isDivisibleBy1987(n: Int): Boolean {
532            return n % 1987 == 0
533        }
534
535        private fun isDivisibleBy2011(n: Int): Boolean {
536            return n % 2011 == 0
537        }
538
539        private fun isDivisibleBy2047(n: Int): Boolean {
540            return n % 2047 == 0
541        }
542
543        private fun isDivisibleBy2077(n: Int): Boolean {
544            return n % 2077 == 0
545        }
546
547        private fun isDivisibleBy2101(n: Int): Boolean {
548            return n % 2101 == 0
549        }
550
551        private fun isDivisibleBy2131(n: Int): Boolean {
552            return n % 2131 == 0
553        }
554
555        private fun isDivisibleBy2161(n: Int): Boolean {
556            return n % 2161 == 0
557        }
558
559        private fun isDivisibleBy2191(n: Int): Boolean {
560            return n % 2191 == 0
561        }
562
563        private fun isDivisibleBy2221(n: Int): Boolean {
564            return n % 2221 == 0
565        }
566
567        private fun isDivisibleBy2251(n: Int): Boolean {
568            return n % 2251 == 0
569        }
570
571        private fun isDivisibleBy2281(n: Int): Boolean {
572            return n % 2281 == 0
573        }
574
575        private fun isDivisibleBy2311(n: Int): Boolean {
576            return n % 2311 == 0
577        }
578
579        private fun isDivisibleBy2341(n: Int): Boolean {
580            return n % 2341 == 0
581        }
582
583        private fun isDivisibleBy2371(n: Int): Boolean {
584            return n % 2371 == 0
585        }
586
587        private fun isDivisibleBy2401(n: Int): Boolean {
588            return n % 2401 == 0
589        }
590
591        private fun isDivisibleBy2431(n: Int): Boolean {
592            return n % 2431 == 0
593        }
594
595        private fun isDivisibleBy2461(n: Int): Boolean {
596            return n % 2461 == 0
597        }
598
599        private fun isDivisibleBy2491(n: Int): Boolean {
600            return n % 2491 == 0
601        }
602
603        private fun isDivisibleBy2521(n: Int): Boolean {
604            return n % 2521 == 0
605        }
606
607        private fun isDivisibleBy2551(n: Int): Boolean {
608            return n % 2551 == 0
609        }
610
611        private fun isDivisibleBy2581(n: Int): Boolean {
612            return n % 2581 == 0
613        }
614
615        private fun isDivisibleBy2611(n: Int): Boolean {
616            return n % 2611 == 0
617        }
618
619        private fun isDivisibleBy2641(n: Int): Boolean {
620            return n % 2641 == 0
621        }
622
623        private fun isDivisibleBy2671(n: Int): Boolean {
624            return n % 2671 == 0
625        }
626
627        private fun isDivisibleBy2701(n: Int): Boolean {
628            return n % 2701 == 0
629        }
630
631        private fun isDivisibleBy2731(n: Int): Boolean {
632            return n % 2731 == 0
633        }
634
635        private fun isDivisibleBy2761(n: Int): Boolean {
636            return n % 2761 == 0
637        }
638
639        private fun isDivisibleBy2791(n: Int): Boolean {
640            return n % 2791 == 0
641        }
642
643        private fun isDivisibleBy2821(n: Int): Boolean {
644            return n % 2821 == 0
645        }
646
647        private fun isDivisibleBy2851(n: Int): Boolean {
648            return n % 2851 == 0
649        }
650
651        private fun isDivisibleBy2881(n: Int): Boolean {
652            return n % 2881 == 0
653        }
654
655        private fun isDivisibleBy2911(n: Int): Boolean {
656            return n % 2911 == 0
657        }
658
659        private fun isDivisibleBy2941(n: Int): Boolean {
660            return n % 2941 == 0
661        }
662
663        private fun isDivisibleBy2971(n: Int): Boolean {
664            return n % 2971 == 0
665        }
666
667        private fun isDivisibleBy3001(n: Int): Boolean {
668            return n % 3001 == 0
669        }
670
671        private fun isDivisibleBy3031(n: Int): Boolean {
672            return n % 3031 == 0
673        }
674
675        private fun isDivisibleBy3061(n: Int): Boolean {
676            return n % 3061 == 0
677        }
678
679        private fun isDivisibleBy3091(n: Int): Boolean {
680            return n % 3091 == 0
681        }
682
683        private fun isDivisibleBy3121(n: Int): Boolean {
684            return n % 3121 == 0
685        }
686
687        private fun isDivisibleBy3151(n: Int): Boolean {
688            return n % 3151 == 0
689        }
690
691        private fun isDivisibleBy3181(n: Int): Boolean {
692            return n % 3181 == 0
693        }
694
695        private fun isDivisibleBy3211(n: Int): Boolean {
696            return n % 3211 == 0
697        }
698
699        private fun isDivisibleBy3241(n: Int): Boolean {
699            return n % 3241 == 0
700        }
701
702        private fun isDivisibleBy3271(n: Int): Boolean {
703            return n % 3271 == 0
704        }
705
706        private fun isDivisibleBy3301(n: Int): Boolean {
707            return n % 3301 == 0
708        }
709
710        private fun isDivisibleBy3331(n: Int): Boolean {
711            return n % 3331 == 0
712        }
713
714        private fun isDivisibleBy3361(n: Int): Boolean {
715            return n % 3361 == 0
716        }
717
718        private fun isDivisibleBy3391(n: Int): Boolean {
719            return n % 3391 == 0
720        }
721
722        private fun isDivisibleBy3421(n: Int): Boolean {
723            return n % 3421 == 0
724        }
725
726        private fun isDivisibleBy3451(n: Int): Boolean {
727            return n % 3451 == 0
728        }
729
730        private fun isDivisibleBy3481(n: Int): Boolean {
731            return n % 3481 == 0
732        }
733
734        private fun isDivisibleBy3511(n: Int): Boolean {
735            return n % 3511 == 0
736        }
737
738        private fun isDivisibleBy3541(n: Int): Boolean {
739            return n % 3541 == 0
740        }
741
742        private fun isDivisibleBy3571(n: Int): Boolean {
743            return n % 3571 == 0
744        }
745
746        private fun isDivisibleBy3601(n: Int): Boolean {
747            return n % 3601 == 0
748        }
749
750        private fun isDivisibleBy3631(n: Int): Boolean {
751            return n % 3631 == 0
752        }
753
754        private fun isDivisibleBy3661(n: Int): Boolean {
755            return n % 3661 == 0
756        }
757
758        private fun isDivisibleBy3691(n: Int): Boolean {
759            return n % 3691 == 0
760        }
761
762        private fun isDivisibleBy3721(n: Int): Boolean {
763            return n % 3721 == 0
764        }
765
766        private fun isDivisibleBy3751(n: Int): Boolean {
767            return n % 3751 == 0
768        }
769
770        private fun isDivisibleBy3781(n: Int): Boolean {
771            return n % 3781 == 0
772        }
773
774        private fun isDivisibleBy3811(n: Int): Boolean {
775            return n % 3811 == 0
776        }
777
778        private fun isDivisibleBy3841(n: Int): Boolean {
779            return n % 3841 == 0
780        }
781
782        private fun isDivisibleBy3871(n: Int): Boolean {
783            return n % 3871 == 0
784        }
785
786        private fun isDivisibleBy3901(n: Int): Boolean {
787            return n % 3901 == 0
788        }
789
790        private fun isDivisibleBy3931(n: Int): Boolean {
791            return n % 3931 == 0
792        }
793
794        private fun isDivisibleBy3961(n: Int): Boolean {
795            return n % 3961 == 0
796        }
797
798        private fun isDivisibleBy3991(n: Int): Boolean {
799            return n % 3991 == 0
800        }
801
802        private fun isDivisibleBy4021(n: Int): Boolean {
803            return n % 4021 == 0
804        }
805
806        private fun isDivisibleBy4051(n: Int): Boolean {
807            return n % 4051 == 0
808        }
809
810        private fun isDivisibleBy4081(n: Int): Boolean {
811            return n % 4081 == 0
812        }
813
814        private fun isDivisibleBy4111(n: Int): Boolean {
815            return n % 4111 == 0
816        }
817
818        private fun isDivisibleBy4141(n: Int): Boolean {
819            return n % 4141 == 0
820        }
821
822        private fun isDivisibleBy4171(n: Int): Boolean {
823            return n % 4171 == 0
824        }
825
826        private fun isDivisibleBy4201(n: Int): Boolean {
827            return n % 4201 == 0
828        }
829
830        private fun isDivisibleBy4231(n: Int): Boolean {
831            return n % 4231 == 0
832        }
833
834        private fun isDivisibleBy4261(n: Int): Boolean {
835            return n % 4261 == 0
836        }
837
838        private fun isDivisibleBy4291(n: Int): Boolean {
839            return n % 4291 == 0
840        }
841
842        private fun isDivisibleBy4321(n: Int): Boolean {
843            return n % 4321 == 0
844        }
845
846        private fun isDivisibleBy4351(n: Int): Boolean {
847            return n % 4351 == 0
848        }
849
850        private fun isDivisibleBy4381(n: Int): Boolean {
851            return n % 4381 == 0
852        }
853
854        private fun isDivisibleBy4411(n: Int): Boolean {
855            return n % 4411 == 0
856        }
857
858        private fun isDivisibleBy4441(n: Int): Boolean {
859            return n % 4441 == 0
860        }
861
862        private fun isDivisibleBy4471(n: Int): Boolean {
863            return n % 4471 == 0
864        }
865
866        private fun isDivisibleBy4501(n: Int): Boolean {
867            return n % 4501 == 0
868        }
869
870        private fun isDivisibleBy4531(n: Int): Boolean {
871            return n % 4531 == 0
872        }
873
874        private fun isDivisibleBy4561(n: Int): Boolean {
875            return n % 4561 == 0
876        }
877
878        private fun isDivisibleBy4591(n: Int): Boolean {
879            return n % 4591 == 0
880        }
881
882        private fun isDivisibleBy4621(n: Int): Boolean {
883            return n % 4621 == 0
884        }
885
886        private fun isDivisibleBy4651(n: Int): Boolean {
887            return n % 4651 == 0
888        }
889
890        private fun isDivisibleBy4681(n: Int): Boolean {
891            return n % 4681 == 0
892        }
893
894        private fun isDivisibleBy4711(n: Int): Boolean {
895            return n % 4711 == 0
896        }
897
898        private fun isDivisibleBy4741(n: Int): Boolean {
899            return n % 4741 == 0
900        }
901
902        private fun isDivisibleBy4771(n: Int): Boolean {
903            return n % 4771 == 0
904        }
905
906        private fun isDivisibleBy4801(n: Int): Boolean {
907            return n % 4801 == 0
908        }
909
910        private fun isDivisibleBy4831(n: Int): Boolean {
911            return n % 4831 == 0
912        }
913
914        private fun isDivisibleBy4861(n: Int): Boolean {
915            return n % 4861 == 0
916        }
917
918        private fun isDivisibleBy4891(n: Int): Boolean {
919            return n % 4891 == 0
920        }
921
922        private fun isDivisibleBy4921(n: Int): Boolean {
923            return n % 4921 == 0
924        }
925
926        private fun isDivisibleBy4951(n: Int): Boolean {
927            return n % 4951 == 0
928        }
929
930        private fun isDivisibleBy4981(n: Int): Boolean {
931            return n % 4981 == 0
932        }
933
934        private fun isDivisibleBy5011(n: Int): Boolean {
935            return n % 5011 == 0
936        }
937
938        private fun isDivisibleBy5041(n: Int): Boolean {
939            return n % 5041 == 0
940        }
941
942        private fun isDivisibleBy5071(n: Int): Boolean {
943            return n % 5071 == 0
944        }
945
946        private fun isDivisibleBy5101(n: Int): Boolean {
947            return n % 5101 == 0
948        }
949
950        private fun isDivisibleBy5131(n: Int): Boolean {
951            return n % 5131 == 0
952        }
953
954        private fun isDivisibleBy5161(n: Int): Boolean {
955            return n % 5161 == 0
956        }
957
958        private fun isDivisibleBy5191(n: Int): Boolean {
959            return n % 5191 == 0
960        }
961
962        private fun isDivisibleBy5221(n: Int): Boolean {
963            return n % 5221 == 0
964        }
965
966        private fun isDivisibleBy5251(n: Int): Boolean {
967            return n % 5251 == 0
968        }
969
970        private fun isDivisibleBy5281(n: Int): Boolean {
971            return n % 5281 == 0
972        }
973
974        private fun isDivisibleBy5311(n: Int): Boolean {
975            return n % 5311 == 0
976        }
977
978        private fun isDivisibleBy5341(n: Int): Boolean {
979            return n % 5341 == 0
980        }
981
982        private fun isDivisibleBy5371(n: Int): Boolean {
983            return n % 5371 == 0
984        }
985
986        private fun isDivisibleBy5401(n: Int): Boolean {
987            return n % 5401 == 0
988        }
989
990        private fun isDivisibleBy5431(n: Int): Boolean {
991            return n % 5431 == 0
992        }
993
994        private fun isDivisibleBy5461(n: Int): Boolean {
995            return n % 5461 == 0
996        }
997
998        private fun isDivisibleBy5491(n: Int): Boolean {
999            return n % 5491 == 0
1000        }
1001
1002        private fun isDivisibleBy5521(n: Int): Boolean {
1003            return n % 5521 == 0
1004        }
1005
1006        private fun isDivisibleBy5551(n: Int): Boolean {
1007            return n % 5551 == 0
1008        }
1009
1010        private fun isDivisibleBy5581(n: Int): Boolean {
1011            return n % 5581 == 0
1012        }
1013
1014        private fun isDivisibleBy5611(n: Int): Boolean {
1015            return n % 5611 == 0
1016        }
1017
1018        private fun isDivisibleBy5641(n: Int): Boolean {
1019            return n % 5641 == 0
1020        }
1021
1022        private fun isDivisibleBy5671(n: Int): Boolean {
1023            return n % 5671 == 0
1024        }
1025
1026        private fun isDivisibleBy5701(n: Int): Boolean {
1027            return n % 5701 == 0
1028        }
1029
1030        private fun isDivisibleBy5731(n: Int): Boolean {
1031            return n %
```

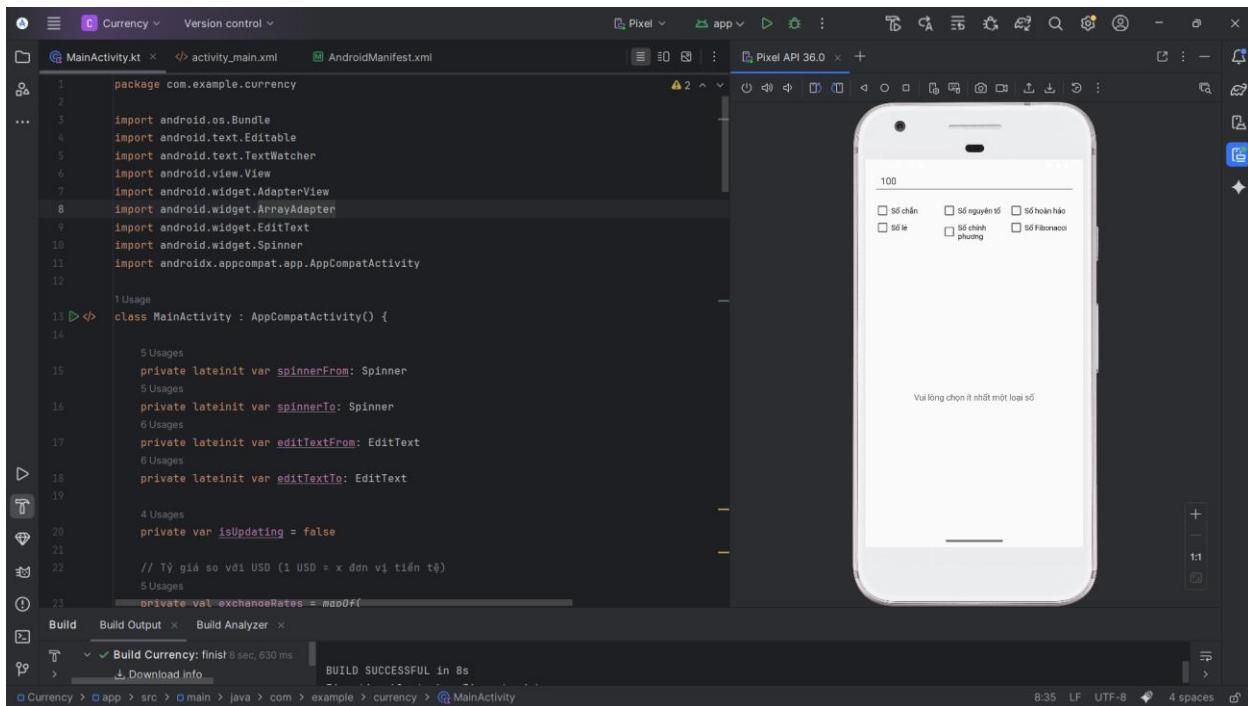
## Số Fibonacci



The screenshot shows the Android Studio interface with the project 'NumberList' open. The code editor displays the MainActivity.kt file, which contains Java code for checking if a number is perfect, square, or Fibonacci. The right side of the screen shows a mobile device emulator running the app, displaying a list of numbers from 1 to 89. A checkbox for 'Số Fibonacci' is checked in the filter bar at the top of the emulator screen.

```
10     class MainActivity : AppCompatActivity() {
11         private fun isPerfect(n: Int): Boolean {
12             var sum = 1
13             for (i in 2..n/2) {
14                 if (i * i == n) {
15                     sum += i
16                 }
17             }
18             return sum == n
19         }
20
21         private fun isSquare(n: Int): Boolean {
22             if (n < 1) return false
23             val sqrt = Math.sqrt(n.toDouble()).toI
24             return sqrt * sqrt == n
25         }
26
27         private fun isFibonacci(n: Int): Boolean {
28             if (n < 1) return false
29             var a = 1
30             var b = 1
31
32             while (b < n) {
33                 val temp = a + b
34                 a = b
35                 b = temp
36             }
37             return b == n
38         }
39     }
```

## Không chọn loại số



The screenshot shows the Android Studio interface with the project 'Currency' open. The code editor displays the MainActivity.kt file, which includes code for handling currency conversion between USD and VND. The right side of the screen shows a mobile device emulator running the app, displaying an error message: 'Vui lòng chọn ít nhất một loại số' (Please select at least one type of number). The filter bar at the top of the emulator screen has all checkboxes unchecked.

```
1 package com.example.currency
2
3 import android.os.Bundle
4 import android.text.Editable
5 import android.text.TextWatcher
6 import android.view.View
7 import android.widget.AdapterView
8 import android.widget.ArrayAdapter
9 import android.widget.EditText
10 import android.widget.Spinner
11 import androidx.appcompat.app.AppCompatActivity
12
13 class MainActivity : AppCompatActivity() {
14
15     private lateinit var spinnerFrom: Spinner
16     private lateinit var spinnerTo: Spinner
17     private lateinit var editTextFrom: EditText
18     private lateinit var editTextTo: EditText
19
20     private var isUpdating = false
21
22     // Tỷ giá so với USD (1 USD = x đơn vị tiền tệ)
23     private val exchangeRates = mapOf(
24         "USD" to 23000.0,
25         "EUR" to 26500.0,
26         "JPY" to 200.0,
27         "AUD" to 15000.0,
28         "NZD" to 14000.0,
29         "SGD" to 16000.0,
30         "MYR" to 5000.0,
31         "INR" to 350.0,
32         "CNY" to 3.5,
33         "VND" to 1000.0
34     )
35 }
```