Activity 2 : Elements of Programming (Part 2)

1. Use and adapt the code PowersOfTwo.java, to print the first 50 powers of 2^N. Include your code as well as the output result.

   OUTPUT

   D:\JAVA\Activity2\Code>javac D:\JAVA\Activity2\Code\PowersOfTwo.java

   D:\JAVA\Activity2\Code>java PowersOfTwo 50

   0.- 1

   1.- 2

   2.- 4

   3.- 8

   4.- 16

   5.- 32

   6.- 64

   7.- 128

   8.- 256

   9.- 512

   10.- 1024

   11.- 2048

   12.- 4096

   13.- 8192

   14.- 16384

   15.- 32768

   16.- 65536

   17.- 131072

   18.- 262144

   19.- 524288

   20.- 1048576

   21.- 2097152

   22.- 4194304

   23.- 8388608

   24.- 16777216

25.- 33554432

26.- 67108864

27.- 134217728

28.- 268435456

29.- 536870912

30.- 1073741824

31.- 2147483648

32.- 4294967296

33.- 8589934592

34.- 17179869184

35.- 34359738368

36.- 68719476736

37.- 137438953472

38.- 274877906944

39.- 549755813888

40.- 1099511627776

41.- 2199023255552

42.- 4398046511104

43.- 8796093022208

44.- 17592186044416

45.- 35184372088832

46.- 70368744177664

47.- 140737488355328

48.- 281474976710656

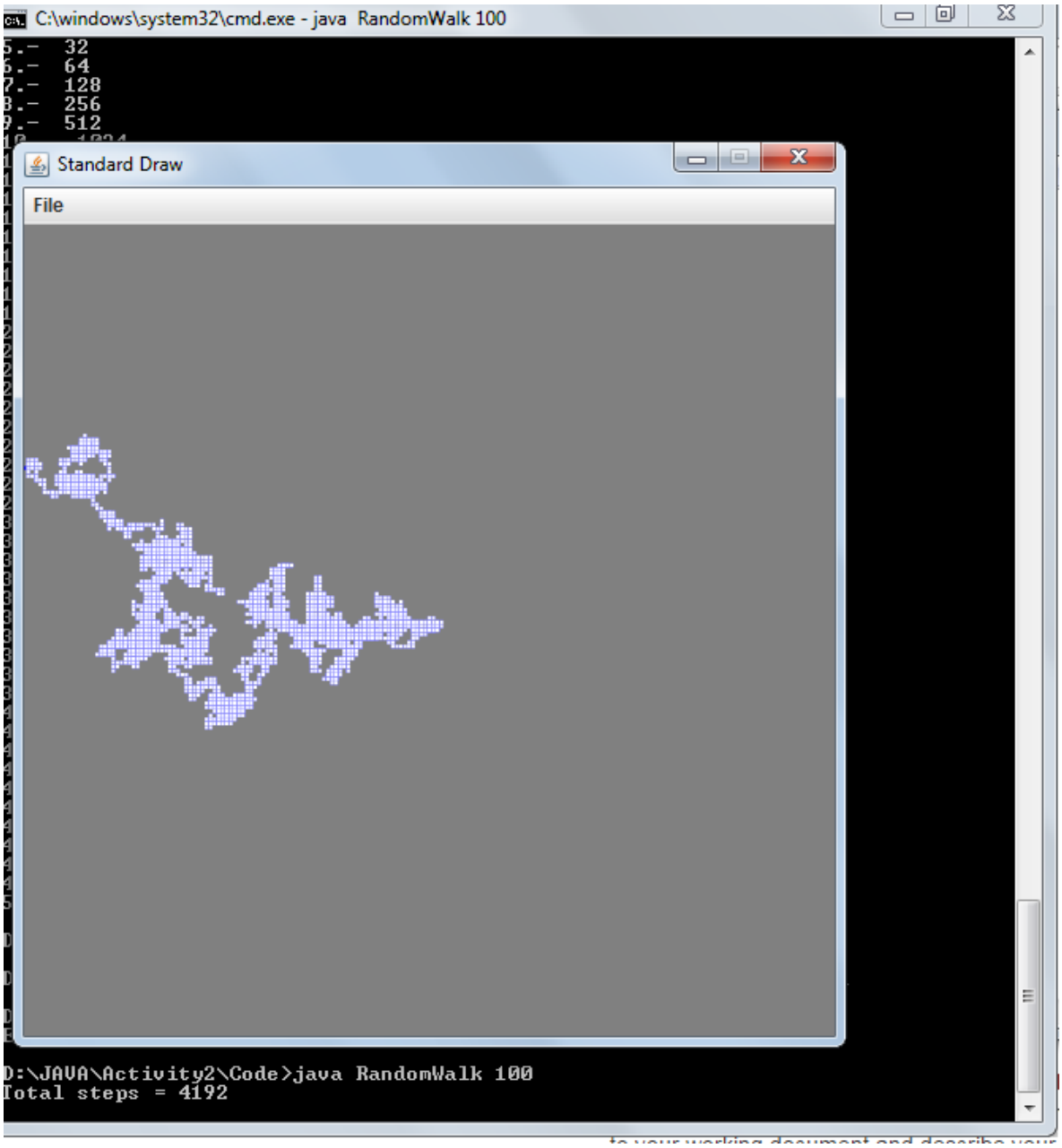
49.- 562949953421312

50.- 1125899906842624

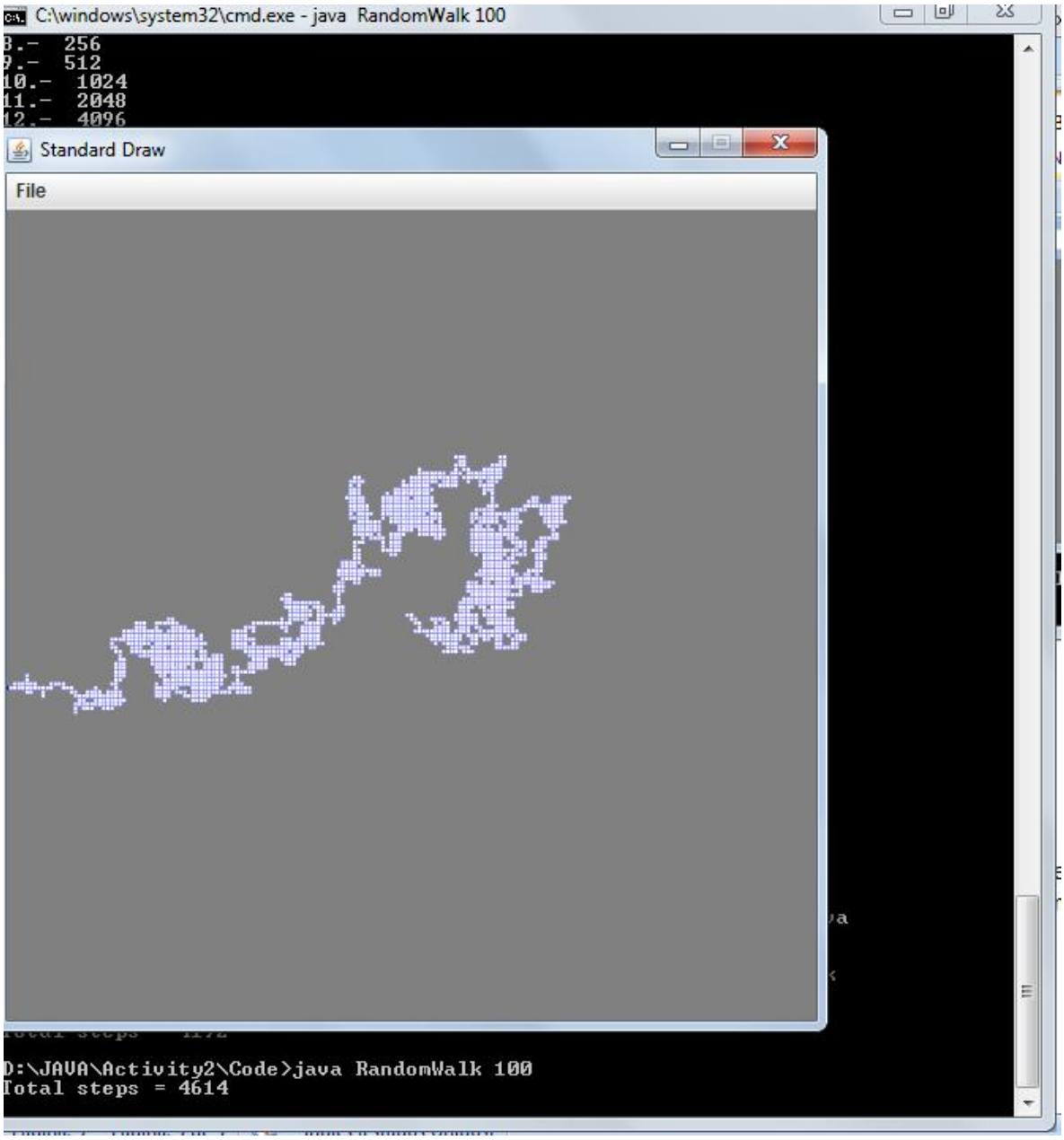
2. Use the code for RandomWalk.java to create 3 pictures that you like, using the number 100 as argument. To compile, you are required to previously compile StdDraw.java. You will produce 3 plots to be copied into your Activity log document.

   OUTPUT

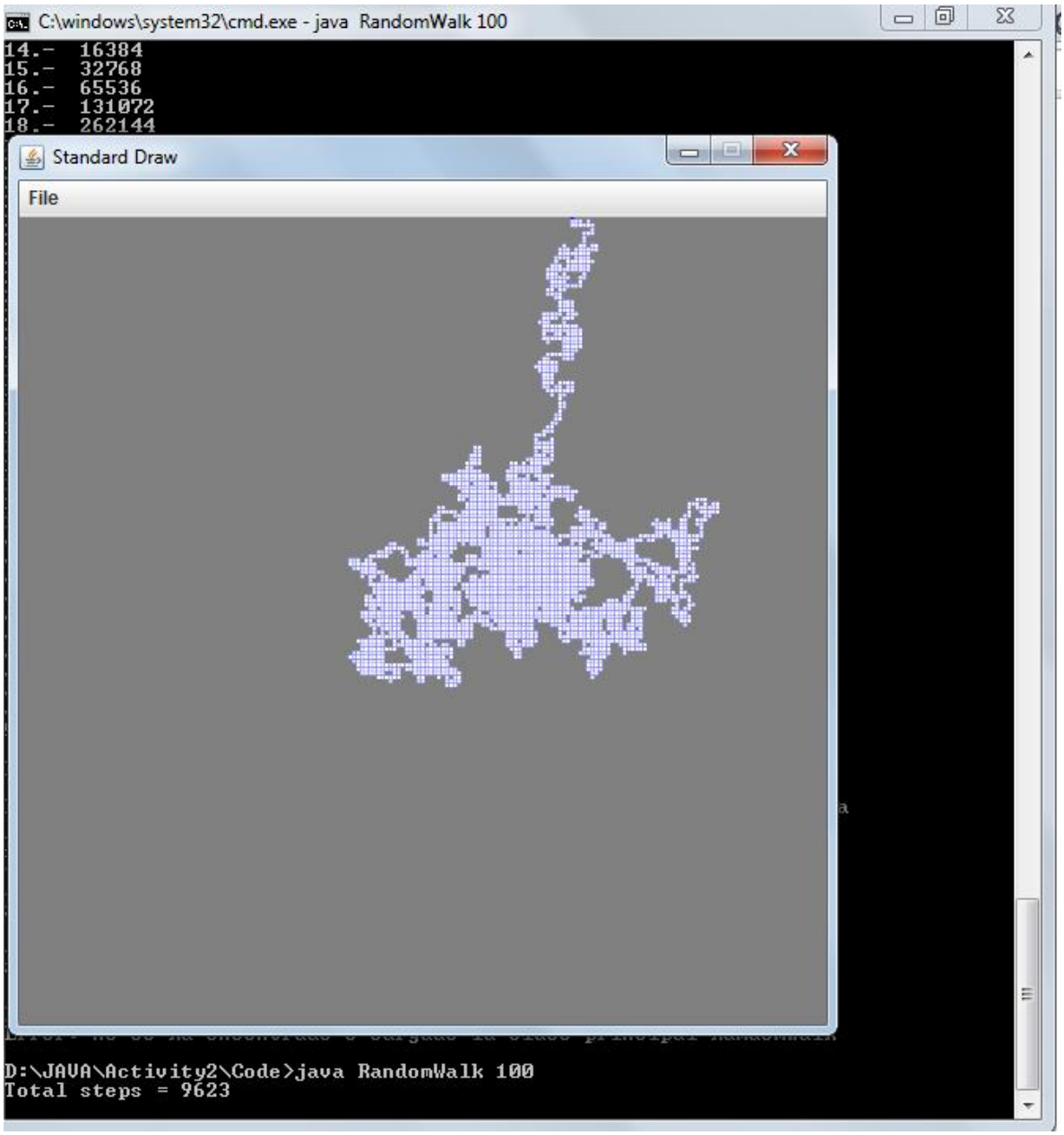
   Random run 1


C:\windows\system32\cmd.exe - java RandomWalk 100
5.- 32
6.- 64
7.- 128
8.- 256
9.- 512
Standard Draw
File
D:\JAVA\Activity2\Code>java RandomWalk 100
Total steps = 4192

Random Run 2


C:\windows\system32\cmd.exe - java RandomWalk 100
8.- 256
9.- 512
10.- 1024
11.- 2048
12.- 4096
Standard Draw
File
D:\JAVA\Activity2\Code>java RandomWalk 100
Total steps = 4614

Random run 3



3. Use the code Factors.java that prints the prime factors of a number. Follow the examples in the code headings comments and you are required to measure the computation time for the next 6 cases: 3, 6, 9, 12, 15, and 18 digit primes

- java Factors 997
- java Factors 999983
- java Factors 999999937
- java Factors 999999999989
- java Factors 999999999999989
- java Factors 9999999999999989

You are free to modify the source code to include a timing function. Here is an example you can review at stackoverflow.com. Include source code and output in your working document.

OUTPUT

D:\JAVA\Activity2\Code>java Factors 997

The prime factorization of 997 is: 997

Elapsed time: 457926 nanoseconds

D:\JAVA\Activity2\Code>java Factors 999983

The prime factorization of 999983 is: 999983

Elapsed time: 229207 nanoseconds

D:\JAVA\Activity2\Code>java Factors 999999937

The prime factorization of 999999937 is: 999999937

Elapsed time: 1820953 nanoseconds

D:\JAVA\Activity2\Code>java Factors 999999999989

The prime factorization of 999999999989 is: 999999999989

Elapsed time: 20479616 nanoseconds

D:\JAVA\Activity2\Code>java Factors 999999999999989

The prime factorization of 999999999999989 is: 999999999999989

Elapsed time: 478634588 nanoseconds

D:\JAVA\Activity2\Code>java Factors 9999999999999989

The prime factorization of 999999999999999989 is: 999999999999999989

Elapsed time: 16611567110 nanoseconds

4. Use the program FunctionGrowth.java that prints a table of the values of *log N*, *N*, *N log N*, N^2, N^3, and 2^N for *N* = 16, 32, 64, ..., 2048. What are the limits of this code? Suppose we want to stop not at N=2048. but at N=1073741824. Modify your code to do this. Add the modified code to your document and include generated output.

OUTPUT

D:\JAVA\Activity2\Code>java FunctionGrowth

log N N N log N N^2 N^3

0 2 1 4 8

1 4 5 16 64

2 8 16 64 512

2 16 44 256 4096

3 32 110 1024 32768

4 64 266 4096 262144

4 128 621 16384 2097152

5 256 1419 65536 16777216

6 512 3194 262144 134217728

6 1024 7097 1048576 1073741824

7 2048 15615 4194304 8589934592

8 4096 34069 16777216 68719476736

9 8192 73817 67108864 549755813888

9 16384 158991 268435456 4398046511104

10 32768 340695 1073741824 35184372088832

11 65536 726817 4294967296 281474976710656

11 131072 1544487 17179869184 2251799813685248

12 262144 3270678 68719476736 18014398509481984

13 524288 6904766 274877906944 144115188075855872

13 1048576 14536349 1099511627776 1152921504606846976

14 2097152 30526334 4398046511104 -9223372036854775808

15 4194304 63959939 17592186044416 0

15 8388608 133734419 70368744177664 0

16 16777216 279097919 281474976710656 0

17 33554432 581453998 1125899906842624 0

18 67108864 1209424316 4503599627370496 0

18 134217728 2147483647 18014398509481984 0

19 268435456 2147483647 72057594037927936 0

20 536870912 2147483647 288230376151711744 0

20 1073741824 2147483647 1152921504606846976 0

5. Modify the code Binary.java that converts any number to binary form, to convert any number to its hexadecimal form. Print the first 256 numbers in hex. Include code and output in your working document.

OUTPUT

D:\JAVA\Activity2\Code>java Binary 256

1 1

2 2

3 3

4 4

5 5

6 6

7 7

8 8

9 9

10 A

11 B

12 C

13 D

14 E

15 F

16 10

17 11

18 12

19 13

20 14

21 15

22 16

23 17

24 18

25 19

26 1A

27 1B

28 1C

29 1D

30 1E

31 1F

32 20

33 21

34 22

35 23

36 24

37 25

38 26

39 27

40 28

41 29

42 2A

43 2B

44 2C

45 2D

46 2E

47 2F

48 30

49 31

50 32

51 33

52 34

53 35

54 36

55 37

56 38

57 39

58 3A

59 3B

60 3C

61 3D

62 3E

63 3F

64 40

65 41

66 42

67 43

68 44

69 45

70 46

71 47

72 48

73 49

74 4A

75 4B

76 4C

77 4D

78 4E

79 4F

80 50

81 51

82 52

83 53

84 54

85 55

86 56

87 57

88 58

89 59

90 5A

91 5B

92 5C

93 5D

94 5E

95 5F

96 60

97 61

98 62

99 63

100 64

101 65

102 66

103 67

104 68

105 69

106 6A

107 6B

108 6C

109 6D

110 6E

111 6F

112 70

113 71

114 72

115 73

116 74

117 75

118 76

119 77

120 78

121 79

122 7A

123 7B

124 7C

125 7D

126 7E

127 7F

128 80

129 81

130 82

131 83

132 84

133 85

134 86

135 87

136 88

137 89

138 8A

139 8B

140 8C

141 8D

142 8E

143 8F

144 90

145 91

146 92

147 93

148 94

149 95

150 96

151 97

152 98

153 99

154 9A

155 9B

156 9C

157 9D

158 9E

159 9F

160 A0

161 A1

162 A2

163 A3

164 A4

165 A5

166 A6

167 A7

168 A8

169 A9

170 AA

171 AB

172 AC

173 AD

174 AE

175 AF

176 B0

177 B1

178 B2

179 B3

180 B4

181 B5

182 B6

183 B7

184 B8

185 B9

186 BA

187 BB

188 BC

189 BD

190 BE

191 BF

192 C0

193 C1

194 C2

195 C3

196 C4

197 C5

198 C6

199 C7

200 C8

201 C9

202 CA

203 CB

204 CC

205 CD

206 CE

207 CF

208 D0

209 D1

210 D2

211 D3

212 D4

213 D5

214 D6

215 D7

216 D8

217 D9

218 DA

219 DB

220 DC

221 DD

222 DE

223 DF

224 E0

225 E1

226 E2

227 E3

228 E4

229 E5

230 E6

231 E7

232 E8

233 E9

234 EA

235 EB

236 EC

237 ED

238 EE

239 EF

240 F0

241 F1

242 F2

243 F3

244 F4

245 F5

246 F6

247 F7

248 F8

249 F9

250 FA

251 FB

252 FC

253 FD

254 FE

255 FF

256 100

6. Modify the code DayOfWeek.java to print the Day of the Week (Sunday, Monday, ...).

   OUTPUT

   D:\JAVA\Activity2\Code>javac D:\JAVA\Activity2\Code\DayOfWeek.java

   D:\JAVA\Activity2\Code>java DayOfWeek 8 2 1953

   Sunday

   D:\JAVA\Activity2\Code>java DayOfWeek 1 1 2000

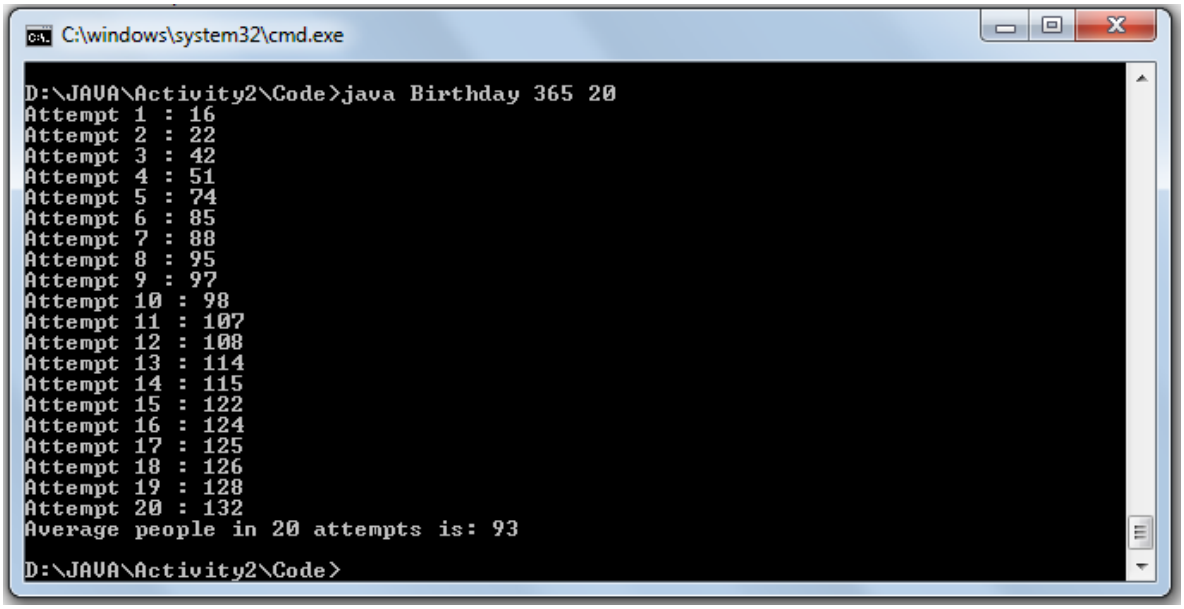
   Saturday

7. Let's play cards. Use the code Deal.java to play 21 or BlackJack for 2 users. You are always the first deal of cards, the house the second. Modify the code to ask for an additional card (Hit=1) or none (Stay=0) for the user. In 20 trials, how many times did you beat the house?. Add the modified code to your working document and describe your experience.

   OUTPUT

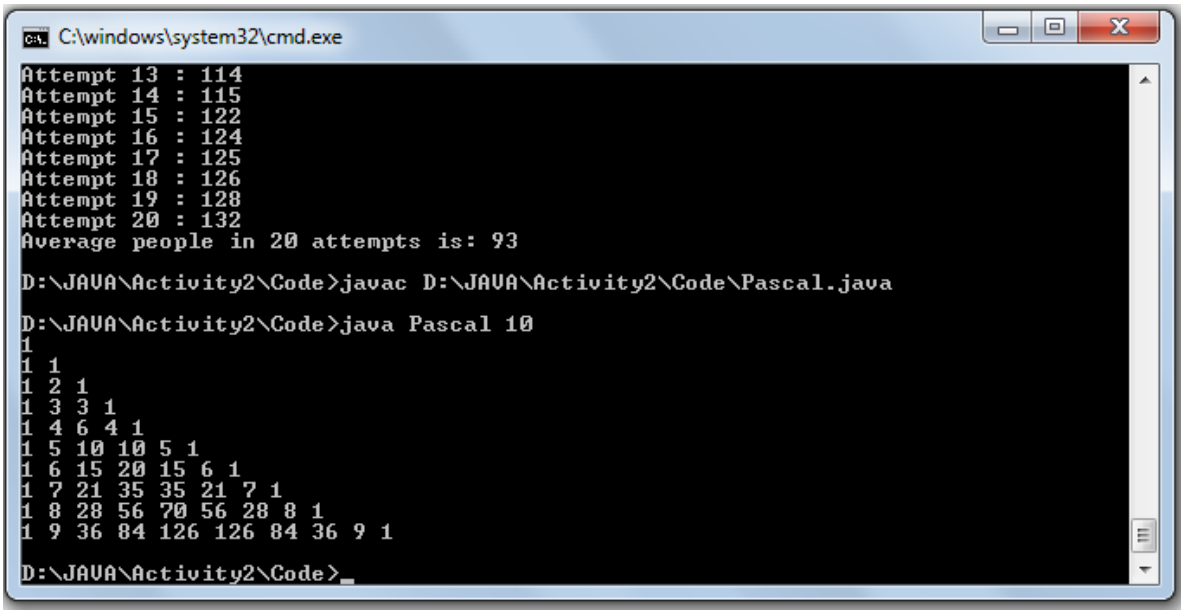
8. Use the code Birthday.java, to run at least 20 experiments and compute the average number of people needed to show up in a room in order that 2 people share the same birthday.

   OUTPUT



9. Use the code to build the Pascal triangle, Pascal.java. Produce a Pascal Triangle to level 10

   OUTPUT



10. You are required to run the code that generates a Sierpinski triangle: Sierpinski.java. This code requires compiling beforehand DrawingPanel.java. Can you guess an algorithm that counts how many solid black inverted triangles and how many upright white triangles per level N. Justify your answer.

    OUTPUT

    D:\JAVA\Activity2\Code>java Sierpinski

What level do you want? 3

Black: 9

White: 4