

2_bigInt.cpp

```

1  /*
2  Program for Question 2: Integer Plot Function (find a smart way to code big integers)
3  We are given a positive integer n. Our task is to print the n using big characters
4  of size 7x7.
5
6  For example, if we get n = 170, our output should be:
7
8      @@  @@@@ @@@@ @@@@
9      @@@  @@  @@  @@
10     @@  @@  @@  @@
11     @@  @@  @@  @@
12     @@  @@  @@  @@
13     @@  @@  @@  @@
14  @@@@@@@@@ @@@  @@@@@@
15
16  My solution is optimal because it maps the ascii art of the numbers to the digits.
17  It converts the number to a string, then loops through the string 7 times to print each row
18  accordingly.
19
20  Run time analysis:
21  T(n) = 7 * # of digits in n = O(n)
22  */
23
24  #include <iostream>
25  #include <string>
26  #include <vector>
27
28  #define BIG_INT_WIDTH 7
29  #define BIG_INT_HEIGHT 7
30
31  std::vector<std::string> zero = {
32      " @@@@@",
33      "@@  @@",
34      "@@  @@",
35      "@@  @@",
36      "@@  @@",
37      "@@  @@",
38      " @@@@@"
39  };
40
41  std::vector<std::string> one = {
42      "  @@ ",
43      " @@@ ",
44      "  @@ ",
45      "  @@ ",
46      "  @@ ",
47      "  @@ ",
48      " @@@@@@"
49  };
50
51  std::vector<std::string> two = {
52      " @@@@@@ ",

```

```

53     "  @@",
54     "  @@",
55     "@@@@@@@@@@",
56     "@@" ,
57     "@@" ,
58     "@@@@@@@@@@"
59 };
60
61 std::vector<std::string> three = {
62     "@@@@@@@@@@",
63     "  @@",
64     "  @@",
65     "  @@@@@@",
66     "  @@",
67     "  @@",
68     "@@@@@@@@@@"
69 };
70
71 std::vector<std::string> four = {
72     "@@  @@",
73     "@@  @@",
74     "@@  @@",
75     "@@@@@@@@@@",
76     "  @@",
77     "  @@",
78     "  @@"
79 };
80
81 std::vector<std::string> five = {
82     "@@@@@@@@@@",
83     "@@" ,
84     "@@" ,
85     "@@@@@@@@@@",
86     "  @@",
87     "  @@",
88     "@@@@@@@@@@"
89 };
90
91 std::vector<std::string> six = {
92     "@@@@@@@@@@",
93     "@@" ,
94     "@@" ,
95     "@@@@@@@@@@",
96     "@@  @@",
97     "@@  @@",
98     "@@@@@@@@@@"
99 };
100
101 std::vector<std::string> seven = {
102     "@@@@@@@@@@@@@@",
103     "  @@",
104     "  @@",
105     "  @@" ,
106     "  @@" ,
107     "  @@" ,

```

```
108     "  @  ",
109 };
110
111 std::vector<std::string> eight = {
112     " @@@@@",
113     "@@  @@",
114     "@@  @@",
115     " @@@@@",
116     "@@  @@",
117     "@@  @@",
118     " @@@@@"
119 };
120
121 std::vector<std::string> nine = {
122     " @@@@@",
123     "@@  @@",
124     "@@  @@",
125     " @@@@@",
126     "  @@",
127     "  @@",
128     " @@@@@"
129 };
130
131 /*
132  * getBigInt(unsigned int n) maps numbers 0 to 9 to their respective
133  * ascii art. If the number is invalid, it just returns a blank string.
134  */
135 std::vector<std::string> getBigInt(unsigned int n) {
136     std::vector<std::string> invalid = {"", ""};
137
138     switch (n)
139     {
140     case 0:
141         return zero;
142         break;
143     case 1:
144         return one;
145         break;
146     case 2:
147         return two;
148         break;
149     case 3:
150         return three;
151         break;
152     case 4:
153         return four;
154         break;
155     case 5:
156         return five;
157         break;
158     case 6:
159         return six;
160         break;
161     case 7:
162         return seven;
```

```
163     break;
164 case 8:
165     return eight;
166     break;
167 case 9:
168     return nine;
169     break;
170 default:
171     return invalid;
172     break;
173 }
174
175 return invalid;
176 }
177
178 void printBigInt(int n) {
179     std::string nString = std::to_string(n); // Convert n to string so that we can loop through each
digit
180
181     for (int i = 0; i < BIG_INT_HEIGHT; i++) { // Print big integer line by line (for loop executes 7
times)
182         for (char& digit : nString) {
183             int mappedValue = digit - '0'; // Converts from ascii to proper decimal value
184             std::cout << getBigInt(mappedValue)[i] << " "; // Print out current line of the corresponding
digit
185         }
186         std::cout << std::endl;
187     }
188 }
189
190 double sec() {
191     return double(clock())/double(CLOCKS_PER_SEC);
192 }
193
194 void timeTestCase(int n) {
195     double T1 = sec();
196
197     std::cout << "n = " << n << ", printBigInt(n) =" << std::endl;
198     printBigInt(n);
199
200     double T2 = sec();
201
202     // std::cout << "Run time of printBigInt(n) repeated " << K << " times: " << T2 - T1 << "s";
203     std::cout << "Run time of printBigInt(n): " << T2 - T1 << "s" << std::endl << std::endl;
204 }
205
206 int main() {
207     timeTestCase(1);
208     timeTestCase(12);
209     timeTestCase(123);
210     timeTestCase(1234);
211     timeTestCase(1234567890);
212     return 0;
213 }
214
```