2_bigInt.cpp

```cpp
/*
  Program for Question 2: Integer Plot Function (find a smart way to code big integers)
  We are given a positive integer n. Our task is to print the n using big characters
  of size 7x7.

  For example, if we get n = 170, our output should be:

     @@   @@@@@@@  @@@@@
    @@@        @@ @@   @@
     @@       @@  @@   @@
     @@      @@   @@   @@
     @@     @@    @@   @@
     @@    @@     @@   @@
  @@@@@@@ @@       @@@@@

  My solution is optimal because it maps the ascii art of the numbers to the digits.
  It converts the number to a string, then loops through the string 7 times to print each row
  accordingly.

  Run time analysis:
  T(n) = 7 * # of digits in n = O(n)
*/

#include <iostream>
#include <string>
#include <vector>

#define BIG_INT_WIDTH 7
#define BIG_INT_HEIGHT 7

std::vector<std::string> zero = {
  " @@@@@ ",
  "@@   @@",
  "@@   @@",
  "@@   @@",
  "@@   @@",
  "@@   @@",
  " @@@@@ "
};

std::vector<std::string> one = {
  "   @@  ",
  "  @@@  ",
  "   @@  ",
  "   @@  ",
  "   @@  ",
  "   @@  ",
  "@@@@@@@"
};

std::vector<std::string> two = {
  "@@@@@@ ",
  "     @@",
  "     @@",
  "@@@@@@@",
  "@@     ",
  "@@     ",
  "@@@@@@@"
};

std::vector<std::string> three = {
  "@@@@@@ ",
  "     @@",
  "     @@",
  "  @@@@ ",
  "     @@",
  "     @@",
  "@@@@@@ "
};

std::vector<std::string> four = {
  "@@   @@",
  "@@   @@",
  "@@   @@",
  " @@@@@@",
  "     @@",
  "     @@",
  "     @@"
};

std::vector<std::string> five = {
  "@@@@@@ ",
  "@@     ",
  "@@     ",
  "@@@@@@ ",
  "     @@",
  "     @@",
  "@@@@@@ "
};

std::vector<std::string> six = {
  " @@@@@ ",
  "@@     ",
  "@@     ",
  "@@@@@@ ",
  "@@   @@",
  "@@   @@",
  " @@@@@ "
};

std::vector<std::string> seven = {
  "@@@@@@@",
  "     @@",
  "    @@ ",
  "   @@  ",
  "  @@   ",
  " @@    ",
  "@@     ",
};

std::vector<std::string> eight = {
  " @@@@@ ",
  "@@   @@",
  "@@   @@",
  " @@@@@ ",
  "@@   @@",
  "@@   @@",
  " @@@@@ "
};

std::vector<std::string> nine = {
  " @@@@@ ",
  "@@   @@",
  "@@   @@",
  " @@@@@@",
  "     @@",
  "     @@",
  " @@@@@ "
};

/*
 * getBigInt(unsigned int n) maps numbers 0 to 9 to their respective
 * ascii art. If the number is invalid, it just returns a blank string.
*/
std::vector<std::string> getBigInt(unsigned int n) {
  std::vector<std::string> invalid = {{""}};

  switch (n)
  {
  case 0:
    return zero;
    break;
  case 1:
    return one;
    break;
  case 2:
    return two;
    break;
  case 3:
    return three;
    break;
  case 4:
    return four;
    break;
  case 5:
    return five;
    break;
  case 6:
    return six;
    break;
  case 7:
    return seven;
    break;
  case 8:
    return eight;
    break;
  case 9:
    return nine;
    break;
  default:
    return invalid;
    break;
  }

  return invalid;
}

void printBigInt(int n) {
  std::string nString = std::to_string(n);  // Convert n to string so that we can loop through each digit

  for (int i = 0; i < BIG_INT_HEIGHT; i++) {  // Print big integer line by line (for loop executes 7 times)
    for (char& digit : nString) {
      int mappedValue = digit - '0'; // Converts from ascii to proper decimal value
      std::cout << getBigInt(mappedValue)[i] << " ";  // Print out current line of the corresponding digit
    }
    std::cout << std::endl;
  }
}

double sec() {
  return double(clock())/double(CLOCKS_PER_SEC);
}

void timeTestCase(int n) {
  double T1 = sec();

  std::cout << "n = " << n << ", printBigInt(n) =" << std::endl;
  printBigInt(n);

  double T2 = sec();

  // std::cout << "Run time of printBigInt(n) repeated " << K << " times: " << T2 - T1 << "s";
  std::cout << "Run time of printBigInt(n): " << T2 - T1 << "s" << std::endl << std::endl;
}

int main() {
  timeTestCase(1);
  timeTestCase(12);
  timeTestCase(123);
  timeTestCase(1234);
  timeTestCase(1234567890);
  return 0;
}

```