Project 02: Three Problems

Due: May 01 by Midnight

Instructions

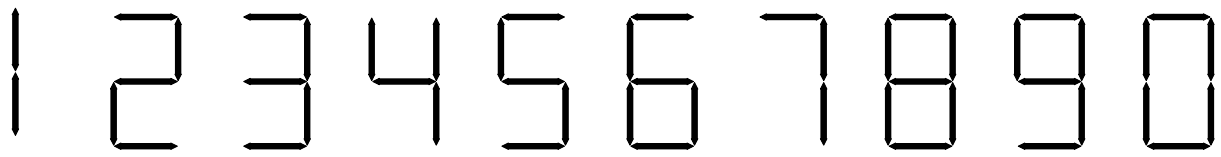
For this project you will be designing and implementing algorithms, in C or C++, to solve problem 1 and problem 2 and Bonus. The program **should output the requirements**.

Also, as a reminder, all of the code for this assignment must be written by you. You may not share code or download solutions off the internet, as doing so will be considered cheating.

Problem Definition:

Problem 01: Sticks (50 points)

Sticks are ideal tools to represent numbers. A common way to represent the ten decimal digits with sticks is the following:



This is identical to how numbers are displayed on an ordinary alarm clock. **With a given number of sticks you can generate a wide range of numbers**. I **am wondering what the smallest and largest numbers can be created by using all your sticks**

Input

- You should prompted from the user **for the number of sticks**. The range of the values, ***n*** should be

  ($2 \leq n \leq 100$)

Output

The smallest and largest numbers you can create with the stick you have. Both numbers should be positive and contain no leading zeroes. (You must try to use all the available sticks)

Sample Test Cases

Sample Input	Sample Output	
	Smallest	Largest
3	7	7
6	6	111
7	8	711
15	108	7111111

Problem 02: Drink Soda (50 points)

Your friend is an absolutely obsessive soda drinker, he simply cannot get enough. Most annoyingly though, he almost never has any money, so **his only obvious legal way to obtain more soda is to recycle empty soda bottles to buy new ones**. **In addition to the empty bottles resulting from his own consumption, he sometimes find empty bottles in the street. One day he was extra thirsty, so he actually drank sodas until he couldn't afford a new one**

Input

- You should prompted user for three non-negative values:
  - **Number of empty soda bottles** in your friend's possession at the start of the day **(*e*)**. The range is

    $e < 1000$
  - **Number of empty soda bottles found during the day (*f*)**. The range is $f < 1000$
  - **Number of empty bottles required to buy a new soda *(C).*** The range is $1 < c < 2000$

Output

How many sodas did your friend drink on his extra thirsty day?

Sample Test Cases

Sample input			Sample output
Possession at the start of the day (e)	**Empty soda bottles found during the day (f)**	**Bottles required to buy a new soda (c)**	
9	**0**	**3**	**4**
5	**5**	**2**	**9**

Bonus Question: Magic Number (20 points)

It's a simple concept. A Magic number **is a number which is evenly divisible by the sum of its digits.**

For example, 24 is a Magic number: the sum of its digits is 2 + 4 = 6 and 24 is divisible by 6.

156 is also a Magic number, since 1 + 5 + 6 = 12 and 156 is divisible by 12.

157 is NOT a Magic number since it is not divisible by 1 + 5 + 7 = 13.

For this problem, you will be given a number n and must find the smallest Magic number >= n

Input

- You should prompted user for a positive integer n ≤ 1 00000

Output

Display the smallest **Magic number greater than or equal to n**.

Sample Test Cases

Sample Input	Sample Output
24	24
25	27
987654	987660

Submission Instruction

All Your programs must adhere to the following constraints:

- The user can input the requisite information how you wish, either by reading from the keyboard, argument lists, or from a file. If read from a file you should use I/O redirection instead of prompting for a filename.
- The user should receive easy to understand output, as to whether their request is possible, and the steps necessary to achieve it if it is possible
- Your code must be well commented
- You must provide a short README.txt file which includes your name and explains how to compile and run your program.
- Additionally, you may write a makefile if you want your code to compile with additional flags.

Submission

A .zip or .rar file containing the following:

1. Separate files named as **problem1, problem2 and bonus**.
2. A README.txt explaining how to compile and run your program

Rubric

Problem01 and Problem 02 worth 50 points each. The breakdown of those points is as follows.

- 40 points: Code satisfies requirements
- 10 points: Professional coding style
    - 5 points: Adequate comments
    - 3 points: Modularity
    - 2 points: Readability
- If your code fails to compile on the CSE machines you may not receive credit for the programming portion ofthe assignment. I recommend not making changes to your code without checking for compilation before you submit.

Bonus Rubric

- 20 points if the code satisfy bonus question