Assignment 4 - Maya Numbers

- Submit all the deliverables to CourSys: https://coursys.sfu.ca
- This assignment may be done **individually**, or **in pairs (2 people)**. Do not show your code to students outside your group, do not copy code found online or from previous course offerings.
- Please direct all questions to the instructor or TA(s) via Discord or in person; do not post questions about the assignment to other online forums.
- See the marking guide for details on how each part will be marked.

1. Binary and Hex Numbers

Complete each of the conversions listed below. You must show your work, not just the answer. You can use Google to check some answers: search “105 in hex” returns “105 = 0x69”).

You can type your work out on the computer, or write it out by hand (must be legible!). In either case, you must generate a PDF to submit to CourSys. If you do the work by hand, you may need to take a photo/scan of it, and convert to a PDF; possibly using Word/LibreOffice Writer to convert to a PDF.

The exact format of how you show your work is unimportant; you just need to demonstrate you did the work vs just using a calculator/Google/program. You are allowed to use a calculator for the individual calculation steps, such as 12 * 16 = ???.

Convert the following from...

1. Decimal to unsigned hexadecimal:
   a) 239
2. Decimal to unsigned 8-bit binary:
   a) 5
   b) 175
3. Hexadecimal to binary (need not show work):
   a) 0x8e
   b) 0x7B41 09FC
4. Binary to hexadecimal (all unsigned values; no need to show work)
   a) 1101
   b) 0110 1111
   c) 0001 1001 1100 0011 1011 0111 1010 0000
5. Decimal to 2's complement binary, 8-bit notation:
   a) 124
   b) -7
   c) -110
6. 2's complement binary notation to decimal:
   a) 1101 0010
   b) 0110 1101
   c) 0000 0100
   d) 1011 0101
   e) 1111 1111 1111 1111 1111 1111 1111 1111
   (hint: Think first! No need to show work for part e.)

 © John Edgar, Brian Fraser

2. Mayan Numbers[1]

First we'll create a function which prints the ancient Maya equivalent of a decimal number (part 2). We'll then use this function in a simple game (part 3). Finally, we'll have the program print out a specific test value (part 4).

Introduction

The Maya number system was a *vigesimal* system, that is base-20. It used symbols to represent numbers, dots (one), lines (five) and a shell for zero. 12_{10} was represented by this symbol,  two lines for ten (5 + 5) and two dots for two (one + one).

Just like decimal, binary or hexadecimal the place values are powers of the number base; 20, in this case. So, symbols represented the number of 1s, 20s, 400s, 8,000s and so on. Unlike our number system, the symbols for each place value were stacked vertically rather than horizontally, with the least significant symbol (the ones) on the bottom.

There is a good Wikipedia article on Maya numerals, which I would suggest reviewing before starting this assignment. It is easy to find more about the subject on the web. If you do some research please ignore anything to do with the priest caste numeral system – they had a separate mixed-base numeral system for dealing with the 360-day Maya calendar!

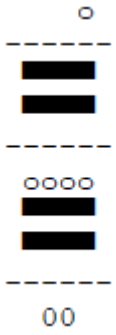
Description

Create a new file for your C++ code in this assignment. Perhaps call it `mayan.cpp`?

You are to write a function that prints the Maya representation of its non-negative decimal integer parameter. Your function must be named `printMayanNumber`, must have a single integer parameter and must return nothing (have `void` as its return type). You will likely want to have multiple extra functions to break down the task of printing out the numbers!

You may assume the number to print is greater than or equal to 0, and strictly less than 1,280,000,000 (= 20^7).

On the right is my program's representation of the decimal number 12,280. There are four place values, which, for clarity, are separated by thin lines (underscores or dashes). 12,280 is made up of one 8,000 (the top dot – a lower case O), ten 400s, fourteen twenties and zero ones (I couldn't figure out how to draw a shell in ASCII so used two zeros to represent zero). You may be wondering how I got those thick black lines to print, see *Printing Lines* section below.



1 Created by John Edgar; modified by Brian Fraser.

 © John Edgar, Brian Fraser

Notes and Advice

- It is intended that you refer to the notes on representation to answer this question. The notes do not cover Maya numerals but they are relevant to this problem.
- Your solution should should be built out of multiple functions. The problem lends itself to decomposition into sub-problems: printing each digit, printing the ones, printing the fives, printing the separator character and so on.
- It is up to you what to name the functions that are called by your `printMayanNumber` function. Though, you should, of course, give them sensible names – and will lose marks if you do not.
- You may include the `cmath` library and may use its `pow()` function. You may **not** use other functions from `cmath`. You may use `iostream` and `iomanip`. You must write the code to convert the number from base 10 to base 20.

Printing Lines

One way of printing a thick block character which – if repeated – can be turned into a thick black line is to print the Unicode character: "\u2580".

To print, it define a character constant:

```
const char* ch = "\u2580";
cout << ch << endl;
```

You must include the * (but don't worry about why yet) and then print it as normal in a `cout` statement.

Unfortunately, exactly what character codes are recognized is a complicated subject and is somewhat dependent on the operating system. To get this to work you may also need to do some additional work.

- Linux and Mac – nothing extra, just print the character
- Windows Subsystem for Linux – nothing extra, just print the character
  (this is the recommended approach under Windows)
- Windows – You can just print a "=" for each character in the line if you like (fast and easy, no loss of marks). However, if you want to get the fancy character, before you run your program, in the VS Code terminal you may need to run run the command:
  `chcp 65001`

When marking, the TA will run your program on a system capable of displaying the character.

 © John Edgar, Brian Fraser

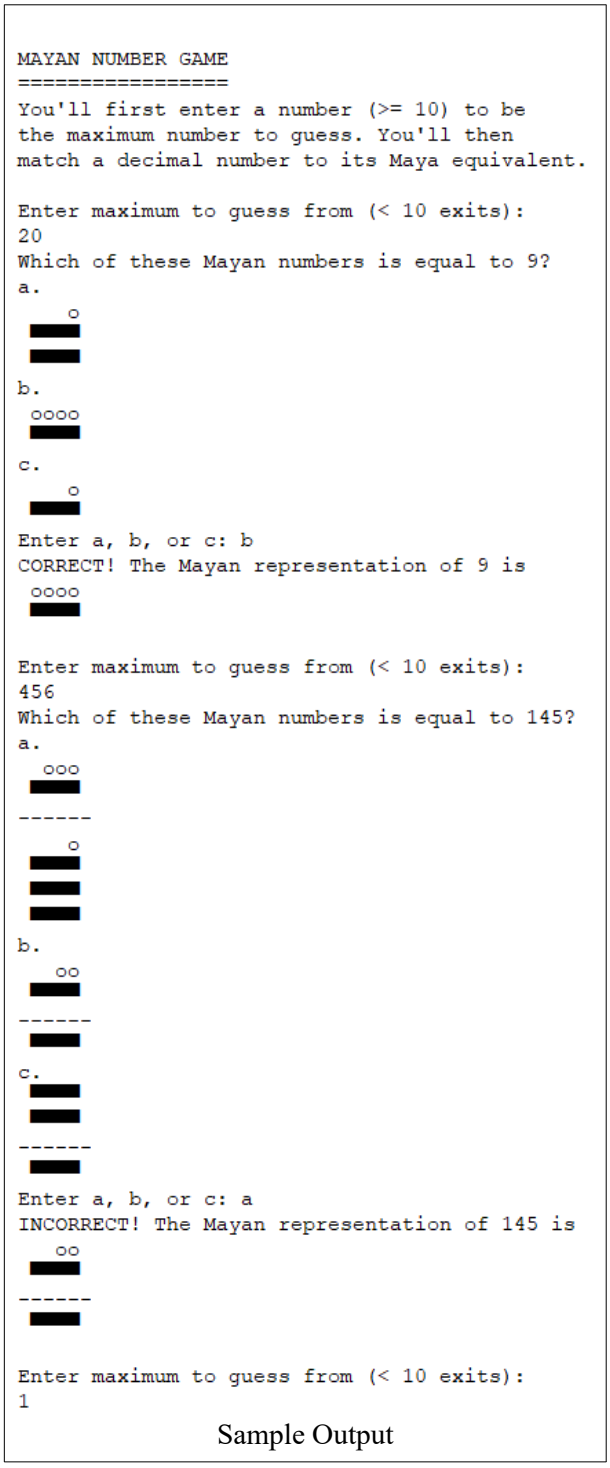
3. Mayan Number Game

Description

Write a game that prints three Maya numbers and asks the user to determine which one matches a decimal number. One execution is shown to the right.

Detailed Requirements

1. The user should be prompted to enter an integer that is at least 10; if a number less than 10 is entered the program should terminate.
2. The game should generate a *correct* answer that is a random value between 4 and the number entered by the user
3. Two incorrect answers should be generated randomly; they should be between
   (*correct* answer * 0.5)
   and (*correct* answer * 1.5);
   for example, if the *correct* answer is 4,000 the incorrect answers should be between 2,000 and 6,000 inclusive.
4. The answers should be labeled *a*, *b* and *c*
5. The three answers should be shown in a random order – it would too easy for the user if the correct answer was always number *a*!
6. The user should be repeatedly prompted to answer by entering *a*, *b* or *c* until one of those values is entered.
7. A message should be printed indicating if the answer is correct or incorrect; in either case the target decimal value and its correct Maya equivalent should be shown.
8. The user should be prompted to play another game, exit if a number less than 10 is entered.



Sample Output

 © John Edgar, Brian Fraser

Notes

1. Like Part 1, it is expected that you solve this problem by writing multiple functions.
    1. You must have a function that accepts as a parameter an integer value and prints that value in Mayan notation to the screen.
    2. Make at least one other function that accepts an argument.
    3. Make at least one function that returns a value. Suggestion: it could be a function that returns a random number in the correct range.
2. You must use at least one for loop.
3. You should include the `ctime` and `cstdlib` libraries (for generating random numbers). *(You may use the more advanced C++ random numbers generators instead, if you want).*
4. You may also include `vector`, `algorithm`, and `random` if needed.
   Hint: One way to randomize the order of the answers is to put them into a vector, and then randomize the order in the vector; however, you may implement it another way.
5. If you have to cast between different numeric types (which you might have to) then use `static_cast` as covered in the presentation on data types

Optional Extra

You may note that there is no requirement for the program to force the incorrect answers to be different from the correct answer. This could lead to a question where the user is forced to choose between the same two – correct – answers and if the "wrong" one is chosen the "you are incorrect" message is displayed. This is obviously a flaw.

The solution is to keep on randomly choosing incorrect answers until they are different from the correct answer (and making them different from each other seems like a good idea as well). You are encouraged to implement this additional functionality – bur for zero marks!

4. Print an Initial Number

At the start of `main()`, before the game plays, print out the Maya number for **2,072,025**.
This will be useful for marking how your program prints out a single number in the Maya system. (This is not shown in the sample output of the previous section).

5. Deliverables

Submit the following to the CourSys: https://coursys.sfu.ca

1. .pdf on the numeric conversions (part 1)
2. .cpp file for the Mayan numbers (parts 2, 3, and 4)

Please remember that all submissions will automatically be compared for unexplainable similarities. This comparison will also include similar assignments from previous semesters and programs on the internet. Please review the notes from lecture on the expectations for academic honesty.

 © John Edgar, Brian Fraser