CS2211a Assignment 5

Issued on: Thursday, November 21, 2013

Due by: Thursday, 11:55 pm, November 28, 2013

- In this assignment, only electronic submission (***attachments***) at owl.uwo.ca is required. Attachments must include:
  - ***ONE pdf*** file that has all sample outputs, and any related communications or discussions for all questions.
  - ***text*** (.c file) soft copy of the programs that you wrote for each question (*one program attachment per question*), i.e., ***TWO*** C program files in total.

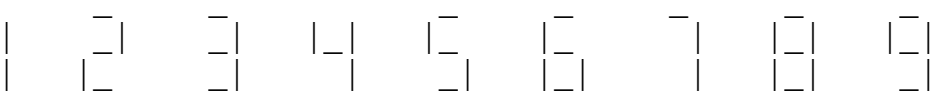
  So, in total, you will submit 1 + 2 = 3 files.

- Late assignments are strongly discouraged
  - 10% will be deducted from a late assignment (up to 24 hours after the due date/time)
  - After 24 hours from the due date/time, late assignments will receive a zero grade.

Copyright © 2013 Mahmoud El-Sakka.

Program 1 (65 marks)

Calculators, watches, and other electronic devices often rely on seven-segment display for numeric output. To form a digit, such devices "*turn on*" some of the seven segments while leaving others "*off*".



Write a program that prompts the user for a number and then displays the number, using '_' and '|' characters to simulate the effect of a seven-segment display. The program should perform as follows:

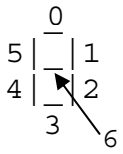
```
Enter a number: 491-9014

    _   _   _
|_| |_|   | |_| | |   | |_|
  |  _|   |  _| |_|   |   |
```

Characters other than digits should be ignored. Write the program so that the maximum number of digits is controlled by a macro named `MAX_DIGITS`, which has the value 20. If the number contains more than this number of digits, *the extra digits are ignored*.

Hints: Use two external arrays.

- `segments[10][7]` array which stores constant data representing the correspondence between digits and seven segments. Order the segments as shown in the figure below:



  The digit 0 will be represented as {`1, 1, 1, 1, 1, 1, 0`}, the digit 1 will be represented as {`0, 1, 1, 0, 0, 0, 0`}, the digit 2 will be represented as {`1, 1, 0, 1, 1, 0, 1`}, the digit 3 will be represented as {`1, 1, 1, 1, 0, 0, 1`}, and so on.

- `digits[3][MAX_DIGITS * 4]` will be an array of characters with 3 rows (since each segmented digit is three characters high) and `MAX_DIGITS * 4` columns (digits are three characters wide, but a space is needed between digits for readability).

RESTRICTION: Other than declaring the arrays, you are not allowed to use the array notation, i.e., you are not allowed to write in your program something like `segments[i][j]` or `digits[i][j]` Instead, you have to use pointers. Failure to do so will cost you most of the question mark.

Write your program as four functions:

1. `main`,
2. `clear_digits_array` which will store blank characters into all elements of the `digits` array
3. `process_digit` which will store the seven-segment representation of a `digit` into a specified position in the `digits` array (positions range from `0` to `MAX_DIGITS - 1`)
4. `print_digits_array` which will display the rows of the `digits` array, each on a single line, producing output such as that shown in the example.

Here are the prototypes for the latter three functions:

```
void clear_digits_array(void);
void process_digit(int digit, int position);
void print_digits_array(void);
```

You should include as many test cases as possible to demonstrate various cases.

Copyright © 2013 Mahmoud El-Sakka.

Program 2 (35 marks)

Declare a tag named `complex_t` for a structure with two members, `real` and `imaginary`, of type `double`.

Write a function that accepts two parameters of type `complex_t` and returns a `complex_t` of their multiplication.

Write a function that accepts two parameters of type **pointer** to `complex_t` and returns a **pointer** to a `complex_t` of their division as a **pointer** to a `complex_t`.

Hint, you will need to ***malloc*** a memory for the returned **pointer** to a `complex_t`.

Note that: if c_1 and c_2 are two complex numbers, where $c_1 = a_1 + j\ b_1$, $c_2 = a_2 + j\ b_2$, and j is a symbol representing $\sqrt{-1}$, then

$$c_1 \times c_2 = (a_1 \times a_2 - b_1 \times b_2) + j\ (a_2 \times b_1 + a_1 \times b_2)$$
$$c_1 \div c_2 = (a_1 \times a_2 + b_1 \times b_2) \div (a_2^2 + b_2^2) + j\ (a_2 \times b_1 - a_1 \times b_2) \div (a_2^2 + b_2^2)$$

- Write a main program that declares four variables of type `complex_t`.
- Prompt the user to input values from the keyboard to initialize two of these structure variables.
- Pass these initialized two structure variables to the two functions described above and store the results in the other two variables.
- Print the returned structure values that indicate the multiplication and the division of the two complex numbers.

If you will declare `complex_t` using *typedef*, show all required changes in your program that need to be done.

You should include as many test cases as possible to demonstrate various cases.

Copyright © 2013 Mahmoud El-Sakka.