Homework 3, CSE 1310

Topics: strings, named constants, meaningful variable names, switch (100 points)

Task 1 (15 pts) – Mad Libs with functions

File: **mad_libs_fcts.c**
Objective: work with functions. Continue practicing strings.

Modify the mad_libs.c program from homework 2 to do the game in one function: `void play_1_mad_libs(void)`

The function play_1_mad)libs implements player 1. It will ask the user for 4 words and print the resulting text. It will use the above text every time.

In the `main()` function call `play_1_mad_libs()` twice so that the user can play the game 2 times.

In the Mad Libs game player 1 has prepared a text like the one below and then asks player 2 to give the missing words, by specifying the category, and they fill in the blanks the given words and then they read the updated text which is often funny.
Our school cafeteria has _____ food. Just thinking about it makes my stomach _____.
(adjective1) (verb1)

The spaghetti is _____ and tastes like _____.
(adjective2) (noun)

Program specifications:

1. Assume the max length of a word entered by the user is 30.
2. Use #define to create a constant that holds the capacity of the char arrays (strings) used in the program.

Sample run 1:

```
Play the FIRST game of Mad Libs:
Enter an adjective (all uppercase): MURKY
Enter a verb (all uppercase): SWIM
Enter an adjective (all uppercase): GREEN
Enter a noun (all uppercase): NEWS
Our school cafeteria has _MURKY_ food. Just thinking about it makes my stomach
_SWIM_. The spaghetti is _GREEN_ and tastes like _NEWS_.

Play the SECOND game of Mad Libs:
Enter an adjective (all uppercase): AWSOME
Enter a verb (all uppercase): FLY
Enter an adjective (all uppercase): BLURY
Enter a noun (all uppercase): STREAM
Our school cafeteria has _AWSOME_ food. Just thinking about it makes
my stomach _FLY_. The spaghetti is _BLURY_ and tastes like _STREAM_.

Bye
```

Task 2 (10 pts) – Happy or sad

File: happy.c
Objective: simple if statement, easy program.

Write a program that will read a character from the user (h or H for happy and s or S for sad) and will print a happy face or a sad face. You have free will in drawing the save. It can be on a single line or on multiple lines.
Hint: you can use a switch with a case for each letter, and have no break between the uppercase and the lowercase versions of the same letter

Sample run 1:	Sample run 2:	Sample run 3:
How are you feeling? **s** … (sad drawing here) Bye	How are you feeling? **H** … (happy drawing here) Bye	How are you feeling? h … (happy drawing here) Bye

Task 3 (20 pts) – Month

File: month.c
Objective: multiple case, longer program.

Write a program that will read an integer between 1 and 12 from the user and will print the corresponding calendar month. If the user gives a number that is too small (e.g. 0) or too large (e.g. 13) the program prints an error message.

Sample run 1:	Sample run 2:	Sample run 3:
Enter month number: **1** January Bye	Enter month number: **3** March Bye	Enter month number: **12** December Bye
Sample run 4:	**Sample run 5:**	**Sample run 6:**
Enter month number: **0** Invalid number Bye	Enter month number: **14** Invalid number Bye	Enter month number: **20** Invalid number Bye

Task 4 (25 pts) – Steak

File: steak.c
Objective: multiple cases.

Write a program that reads a temperature from the user and, using the information from the picture "Steak doneness and levels" from this page, prints the corresponding message:

Rare / Medium Rare / Medium / Medium Well / Well / What have you done?!

For completeness, add message "Start the fire!" when temp is below 120 (this case is not included in the image).

Notes:

1. Note the ambiguity of the information given in the image for the end/start of range values. For example 130 is the end temperature for Rare, but also the start temperature for Medium Rare . I chose to use them as the start value. See for example that for temperature 130 my output is Medium Rare, not Rare. Use my convention:
    a. temp<120 : Start the fire!
    b. 120≤temp<130 : Rare

c. 130≤temp<135 : Medium Rare
d. 135≤temp<145 : Medium
e. 145≤temp<155 : Medium Well
f. 155≤temp<165 : Well
g. 165≤temp : What have you done?!

HINT: use the fact that in if-else if statements, you only get to the next branch if the previous ones failed.

Sample run 1:
`Meat temperature: 130`
`Medium Rare`
	Sample run 4:
	`Meat temperature: 170`
	`What have you done?!`
		Sample run 7:
		`Meat temperature: 145`
		`Medium Well`

Sample run 2:
`Meat temperature: 110`
`Start the fire!`
	Sample run 5:
	`Meat temperature: 120`
	`Rare`	

Sample run 3:
`Meat temperature: 140`
`Medium`
	Sample run 6:
	`Meat temperature: 155`
	`Well`	

Task 5 (30 pts) – Base Conversion B->10

File: digits_B_to_10.c
Objective: use if-statements, longer program, application: base conversion from any base B <= 10 to base 10.

This program will improve the digits program from homework 2 by handling the case when the number given by the user does not have exactly 3 digits. It will also read a base and compute the base conversion from the input number in base B to base 10. If the number given by the user is :

- strictly less than 0 print a message and do not do any processing.
- between 0 and 9 inclusive, print only the ones and the result
- between 10 and 99 inclusive, print the ones and the tens and the result
- between 100 and 999 inclusive, print the ones, the tens and the hundreds and the result
- strictly larger than 999, print a message and do not do any processing.

HINT: use the fact that in if-else if statements, you only get to the next branch if the previous ones failed.

Sample run 1:
`Enter N: 317`
`Enter base B: 8`
`ones : 7`
`tens : 1`
`hundreds : 3`
`Number in base 10: 207`
`Bye`
	Sample run 2:
	`Enter N: 8`
	`Enter base B: 9`
	`ones : 8`
	`Number in base 10: 8`
	`Bye`
		Sample run 3:
		`Enter N: 52`
		`Enter base B: 7`
		`ones : 2`
		`tens : 5`
		`Number in base 10: 37`
		`Bye`

Sample run 4:
`Enter N: 4825`
`This number has more than 3 digits.`
`Bye`
	Sample run 5:
	`Enter N: -10`
	`This number is negative.`
	`Bye`	

Submission and Penalties

All solution files should be placed in a folder called **hw3_Lastname**. Zip this folder and submit it to Canvas.

Penalties

1. ****** Code using elements we had not covered at the time the homework was due, receives no credit.**
2. Code must compile and run.
3. Files for each task must be named as shown.
4. Up to **10** points will be lost for non-compliance with the submission requirements: folder name, all files in a folder, zipped folder, the compressed file is a zip, the program files have extension
5. Each program must have:
    - description and your name at the top
    - proper indentation
    - meaningful variable names