Assignment 2: hangmAnk

Marked out of 100, 9% weight

Topics covered: Nested if, switch, while / for loops, user-defined functions, 1D arrays and strings

1.0 The task

In this assignment, you are required to write functions that help build a hangman game that works with digits only – just the 10 digits that occur in the decimal system (0 – 9). But there is more to it. For this assignment, you will enter the input string and computer will play hangman using random numbers. More on the game is explained later. A sample scenario is also presented in section 2.0. The scenario shows that an input string that may consist of spaces (hint: use fgets to input the string) is first read, some statistics on the input string (details given later) is created and displayed, and then hangman using digits is played. Your code must write definitions for the given prototypes. You will be tested on individual functions – note that you must test all the functions by writing a main. A complete scenario for input string "Aug 18, 2002" is given in section 2.0. You may define additional functions if you wish.

Important tips:

1. Read ahead Zybooks 6.1, 6.2, 6.3 and 6.9 to get started early on the assignment.
2. Start by writing code for function readInput.
3. Read the given scenario in section 2.0 – will help you understand the game and code it.
4. Do not change the prototypes of the functions – we will test your code using these prototypes. You may add more function definitions and use them in your code.
5. DO NOT use global variables. Use of any global variables will result in automatic zero.
6. DO NOT use goto statements. Use of any goto statements will result in automatic zero.
7. Start the assignment early.
8. Make sure that you leave enough time for submitting the assignment on moodle on time.

Defined constants:

```
#define MAX 100
#define MAXD 10
```

- Note that Zybook section 2.23 has a section on using #define.
- MAX defines the maximum number of characters that input string can have.
- MAXD defines the total number of digits

Function prototypes and description:

1	**Prototype:** `void readInput (char str [MAX]);` **Description:** Returns void Reads a string from the keyboard and stores it in str Note that the input string may have spaces (e.g. Oct 20th, 2021). Hint: use fgets instead of scanf

2	**Prototype:**
	`void findStats (char str [MAX],`
	`int * countSpecialChars,`
	`int * countUpperCase,`
	`int * countLowerCase,`
	`int * countNumbers);`
	
	Description:
	Returns void
	This function reads a string and outputs (using call-by-reference parameters) the following:
	- Total number of special characters in str (you may assume that comma and spaces are the only special characters used in this assignment)
	- Total number of uppercase characters in str
	- Total number of lowercase characters in str
	- Total number of digits in str
	
	Example: Input string "Aug 18, 2002" has 3 special characters (2 spaces and 1 comma), 1 uppercase character (i.e. A), 2 lowercase ones (i.e. u and g) and 6 individual digits (i.e. 1, 8, 2, 0, 0 and 2).

3	**Prototype:** `int findTotalChancesGiven (char str [MAX]);` **Description:** Returns int This function finds and returns the total number of unique digits in the given string str. **Example:** Total number of unique digits in string "Aug 18, 2002" is 4 (since the unique digits are 1, 8, 0 and 2). This actually defines the total number of chances a player gets – in this example, the player gets 4 chances to make a guess.

4	**Prototype:**
	`int extractDigits (char str[MAX],`
	`int digitsInInputString [MAX]);`
	
	Description:
	Returns int
	This function finds the digits in the given input string `str` and stores them in `digitsInInputString.`
	It returns the total number of digits found.
	
	Example: If the input string is "Aug 18, 2002", then `digitsInInputString` is populated as [1, 8, 2, 0, 0, 2]. The function returns 6.

5	**Prototype:**
	`int checkPlayerMove (int playersGuess,`
	`int digitsInInputString [MAX],`
	`int totalDigits);`

	Description: Returns int This function returns 1 if the player's guess is correct and 0 otherwise. **Example:** If `playersGuess` is 8, `digitsInInputString` is [1, 8, 2, 0, 0, 2] and `totalDigits` is 6, then the function returns 1. But if the players guess is 7, then it returns 0.
6	**Prototype:** `void updateDigits (int playersGuess, int remainingDigits [MAXD]);` **Description:** Returns void Note that each time in the game, when player makes a guess, a list of remaining digits (i.e. the ones that the player has not guessed yet) is displayed (e.g. in the scenario in section 2.0, lines 29, 42, 55 show the list of remaining digits). This list is maintained in an array called `remainingDigits` that has MAXD elements (note that MAXD is defined as 10). This function updates the array remainingDigits by storing a -1 in the element that represents playersGuess. **Example:** if playersGuess is 2, then this function stores -1 at index 2 of remainingDigits (i.e. remainingDigits [2] = -1)
7	**Prototype:** `void displayDigits (int remainingDigits [MAXD]);` **Description:** Returns void As explained in the previous function, before player makes a guess, a list of remaining digits (i.e. the ones that the player has not guessed yet) is displayed. This function uses `remainingDigits` to display the digits that the player has not guessed so far. The function displays a message followed by all the digits not guessed by the player. **Example:** In the given scenario in section 2.0, lines 29, 42 and 55 are displayed by this function. For example, after the player makes the first guess which is 0, this function displays line 29: **Digits remaining are as follows - player picks one among these: 1 2 3 4 5 6 7 8 9** Note that 0 is not displayed in the above list. Similarly, in the second attempt, when the player guesses 2, this function displays line 42, i.e. **Digits remaining are as follows - player picks one among these: 1 3 4 5 6 7 8 9** Note that both 0 and 2 are not displayed in the above list. since the player has already chosen 0 and 2.
8	**Prototype:** `void displayXs(char exes[MAX], int totalDigits);` **Description:** Returns void Note that in this game, after player makes a guess, the current hangman is displayed in terms of Xs (e.g. lines 16, 25, 38 51, 64 in the given scenario in section 2.0 display hangman). Array exes stores the hangman in terms of Xs or in terms of digits. This function displays the current hangman by displaying

	elements in the array exes. Note that although the size of exes is MAX, the total number of elements that are used by the game is equal to totalDigits. **Example:** If the 2 arguments passed to this function are ['X', 'X', 'X', 'X', 'X', 'X'] and 6, then the hangman displayed is: X X X X X X (as shown on line 16 of the given scenario in section 2.0). Similarly, if the 2 arguments passed to this function are ['X', 'X', 'X', '0', '0', 'X'] and 6, then the hangman displayed is: X X X 0 0 X (as shown on line 25 of the given scenario in section 2.0).
9	**Prototype:** `int countXs (char exes [MAX], int totalDigits);` Description: Returns int This function counts and returns the total number of Xs in the array `exes.` The total number of elements in `exes` is equal to `totalDigits.` **Example:** if exes is ['X', 'X', 'X', '0', '0', 'X'] and totalDigits is 6, then countXs returns 4.

2.0 Sample Scenario:

```
ritu$ ./a.out

Enter a string: 18 Aug,2002

Count of special characters = 2
Count of uppercase characters = 1
Count of lowercase characters = 2
Count of single-digit integers = 6

Press a key to continue

Now the computer plays hangman with numbers - you get 4 chances

Each X depicts a digit in the same order as it occurs in the given string

X X X X X X

Choose a digit among these: 0  1  2  3 4  5  6  7  8  9

Player's choice # 1 is 0
Your choice is correct this time. Well done!

Current hangman scenario is as follws:

 X  X  X  0  0  X

You have 3 guesses left

Digits remaining are as follows - player chooses one among these:   1 2 3 4 5 6 7 8 9

Press a key to continue

Player's choice # 2 is 2
Your choice is correct this time. Well done!

Current hangman scenario is as follws:

 X  X  2  0  0  2

You have 2 guesses left

Digits remaining are as follows - player chooses one among these:   1   3 4 5 6 7 8 9

Press a key to continue

Player's choice # 3 is 1
Your choice is correct this time. Well done!

Current hangman scenario is as follws:

 1  X  2  0  0  2

You have 1 guess left

Digits remaining are as follows - player chooses one among these:       3 4 5 6 7 8 9

Press a key to continue

Player's choice # 4 is 6
Sorry, your choice is incorrect this time. Better luck next time.

Current hangman scenario is as follws:

 1  X  2  0  0  2

You were given 4 chances to play hangman
Number of correct guesses = 3

End of game - better luck next time
```

3.0 Submission Instructions

- Submit a single C file containing your function definitions only (Do not submit main). To submit, upload your C file to the submission box for A2 on moodle. Name your file as lastnameFirstnameA2.c (For example, if Ritu is the first name and Chaturvedi is the last name, the file would be called chaturvediRituA2.c). Incorrect file name will result in penalty.
- Incorrect format of submitted files will result in automatic zero. (Must be a valid .c file)
- The program you submit must compile with no warnings and run successfully for full marks. You get a zero if your program doesn't compile. There is also a penalty for warnings (5% for each unique warning).
- Penalties will occur for missing style, comments, header comments etc.
- DO NOT use global variables. Use of any global variables will result in automatic zero.
- DO NOT use goto statements. Use of any goto statements will result in automatic zero.
- Use the template given below for header comment.
  /!\ Note: The file name, student name and email ID must be changed per student.

```
/************************chaturvediRituA2.c**************

    Student Name: Ritu Chaturvedi Email Id: ritu

    Due Date: October 6th Course Name: CIS 1300

    I have exclusive control over this submission via my password.
    By including this statement in this header comment, I certify that:

    1) I have read and understood the University policy on academic integrity. 2) I
    have completed the Computing with Integrity Tutorial on Moodle; and 3) I have
    achieved at least 80% in the Computing with Integrity Self Test.

    I assert that this work is my own. I have appropriately acknowledged any and
    all material that I have used, whether directly quoted or paraphrased.
    Furthermore, I certify that this assignment was prepared by me specifically for
    this course.

*********************************************************/
```

The program file must contain instructions for the TA on how to compile and run your program in a header comment.

/!\ Note: The file name must be changed per student.

```
/*********************************************************
Compiling the program
The program should be compiled using the following flags: -std=c99 -Wall

compiling:
gcc -std=c99 -Wall chaturvediRituA2.c
```

```
Running: ./a.out
OR

gcc -std=c99 -Wall chaturvediRituA2.c -o assn2

Running the Program: ./assn2
*****************************************************/
```