README

This README contains a technical explanation about how the WOF game has been implemented by group10.

LEFT BLANK ON PURPOSE

Cossa Hugo, Chen Gilles, Gillet Thomas

Index

1. How the Game Works.

There are 3 Puzzles (Words, sentences) and 3 Rounds. At each Round one of the Puzzles will be hidden and the players need to guess consonants or buy vowels to reveal the Puzzle so guessing the entire Puzzle becomes easier. Before they are able to guess, they spin the wheel which can land on:

- Land on Lose Turn -> Lose Turn and next Player continues
- Land on Extra Turn -> Spin Again
- Land on Bankrupt -> Lose Turn and lose all Money from the Current Round
- Land on Money -> Able to now guess a consonant/or a Vowel -> Get Money TIMES the occurrences of the guessed consonant/vowel.

If a guess has been correct, the player will then be allowed to Guess the entire Puzzle. If he decides to do so, he has to enter the Puzzle (lower/upper-case does not matter). If he guesses correctly, the round ends and he is allowed to keep all the money he has earned in the Round, while the other players will end the Turn with 0.

If he decides to not guess the Puzzle, the wheel will be spun again.

A Round ends when the Puzzle has been GUESSED (Enter entire Puzzle) by a player.

If 3 Rounds have been played, the Game ends.

2. The Rules of the Game

You can not buy a vowel on your first turn in a Round.

You can not enter a vowel in the first turn in a Round.

If you land on a wedge containing money value, a correct guess of a letter will grant you the Occurrence of that Letter in the Puzzle * The Money value you landed on.

You can only guess the Puzzle if you are asked if you want to guess the Puzzle.

You don’t have to guess the “whitespaces” inbetween Words in the Puzzle.

After each round the money wedge with the highest value will be increased by 500.

3. Step by Step Terminal Input & Handling

- As soon as you launch the WOF Game, you will be asked to enter the first 3 Puzzles. Note that we have decided that a Puzzle can't be longer than 50 characters! You need to enter the sentences 1 by 1 and confirming each time with ENTER. Your entries will be trimmed of leading and trailing whitespaces. Example (w means whitespace): "wwwwwwheyww" will become "hey".
  "wHello" will become "Hello".

- Now you will be asked to enter the amount of players. The game has been designed to be played by 2 or 3 players. Any INTEGER confirmed by Enter that is not 2 or 3, will be ignored and you will be prompted for the amount again. Any decimal number (e.g. 3,9) will be accepted if the number before the decimal point contains ONLY 2 or 3.
  Any character (unless 2 or 3) will be ignored and prompt you to enter the amount again.
  Any string (e.g. ab!, de/.f) will be ignored unless it leads with 2 or 3 (e.g. 2dfgdfter, 2/, 3f!!2h).

- Now that you have entered the amount of players, you will be asked to enter the Name of the players 1 by 1. Here aswell, we have decided to trim leading and trailing whitespaces.
  Any character/string entered here is accepted. People might have weird names like "X Æ A-12"!

- Now the game can start. First we will print the Puzzle that needs to be guessed. It will be hidden by dashes '-'.
  Example: "The Walking Dead" will be "--- ------- ----"

- A random player will be selected, and after that they will play one after the other.
  Example: 3 Players, Player 2 starts. Therefore, the order will be: (P2->P3->P1->P2->P3)

- The Wheel will be spun as soon as its a players turn. After the animation ends, the Terminal will print what you have landed on. The first 4 '-' in the Wheel are Lose Turn, Bankrupt, Bankrupt, Extra Turn. The following dashes are money values in an ascending fashion.
  Based on what you land the following will happen:
    - Lose Turn: Lose your turn, going to next Player
    - Bankrupt: Lose your turn and all the Money you have gathered in the round.
    - Extra Turn: The wheel will be spun again.
    - Money:
        - Is it the first time you are guessing in this Round?
            - Yes: You will be asked to enter a consonant. Vowels are not accepted, and if Entered, you will be prompted to enter a consonant again. Any character is accepted. Strings are also accepted, but only the first character is considered and therefore your guess.
                - Your consonant is in the Puzzle: You get the Money you landed on * the occurrence of the consonant.
                - Your consonant is not in the Puzzle: You don't get anything and it's the next players turn now.
            - No: You will be asked to enter a consonant OR buy a Vowel.
              Any character is accepted. Strings are also accepted, but only the first character is considered and therefore your guess.
                - Same rules Apply to guessing as above.
                - You buy a vowel contained in the Puzzle: You lose 250 Dollars but get the Money you landed on * the occurrence of the Vowel.
                - You buy a vowel NOT contained in the Puzzle: You lose 250 Dollars and it's the next players turn now.
        - What happens after a successful guess?
            - You will be asked if you want to guess the Puzzle.
                - You want to guess the Puzzle: You now need to enter the entire puzzle. Upper/lower-case does not matter.
                    - You entered the wrong Puzzle: Next Player
                    - You entered the correct Puzzle: You win the round and the round ends.
                - You don't want to guess the Puzzle:
                    - The Wheel will be spun again for you.
        - What happens after a Round ends?
          The person that correctly guessed the Puzzle finishes the Round with the Money he has gathered in that particular Round. Any other people will not have gained any Money in that Round. At the end of each Round, a Standing of the overall Game will be printed.

<u>Example:</u>
Round 2
Tim [700 $]
Josh [2000 $]
Alex [6000 $]

Josh's turn and he guesses the Puzzle correctly.
<u>The standing of Round 2 will be:</u>
Round 2

1. Josh	[2000 $]
2. Tim	[0 $]
2. Alex	[0 $]

The overall Standing of the Game will be: (EXAMPLE, NO MONEY VALUES)
(Highest amount of money overall should be the first)

1. Tim [Accumulated Money from Round 1 + Accumulated Money from Round 2]
2. Josh [Accumulated Money from Round 1 + Accumulated Money from Round 2]
3. Alex [Accumulated Money from Round 1 + Accumulated Money from Round 2]

- What happens if Round 3 ends?
  The Final Standing of the Overall game is printed, and the Person who has accumulated the most amount of money over the course of the 3 Rounds will be the winner.

4. Special “Rules”

- If you don’t have enough money to buy a vowel, but have the option to buy one, you are allowed to go into negative money amount.
- If all consonants have been guessed in the Puzzle, and now it is your first turn in the current Round, you will be allowed to buy a vowel, even though it is your first time playing in the Round.
- If the puzzle has been fully revealed and the turn goes over to you due to the previous player “guessing (pronounciating)” the puzzle in a wrong manner, the Wheel will spin but you will only be allowed to enter your guess for the entire Puzzle since there a no more letters to be revealed. If you guessed correctly, you will receive the amount of money that you landed on, if guessed wrongly, it will be the next players turn.

5.Explanation Main.c

As explained in the section "4. Special Rules", when the puzzle is fully revealed the players don't have the choice to guess consonant or buy vowels. Hence, we had to find a way to skip those steps when the puzzle was indeed revealed.

That's what is done in line 78. The Boolean correctCond is set to false which means that, if the Boolean is false, the while loop (line 83) used for guessing consonant / buying vowel, won't be executed.

We also had to come up with a solution to skip the "Do you want to guess the puzzle?" part.
We thus assign a special value to the occ variable. If it's set to -1, it means that the puzzle is revealed otherwise (value between 0 and X) it represents the number of occurrences of the user guess.

The output asking if the user wants to guess the puzzle at line 100 is therefore skipped thanks to this special case value (occ = -1).

At line 115, we evaluate the code only if the users has found occurrences or if the puzzle is revealed (reason why we test if occ != 0).

At line 118, we only ask the user if he wants to guess the puzzle if the puzzle is not yet revealed.

Since the occ is set to -1, the guess = 'Y'/'N' is not set (because lines 118 to 125 are skipped).
Therefore, at line 127 we ask the user for its puzzle guess if the guess='Y' OR if the occ = -1.

At the 135th line, we used a ternary inside the addMoney() function.
This is following the special rule explained in section "4. Special Rules". If the user solves the puzzle after having guessed a letter or bought a vowel, he will be granted no money. On the other hand, if he guessed the puzzle when this later one was fully revealed (with occ=-1) he will be granted the amount of money he got from the wheel spin.

At line 142, when resetting the players' structures, we used a ternary for the total money computation. If the last value of currentPlayer is equal to i the currentTurnMoney money is added to the player's totalMoney otherwise nothing is added. Indeed, when finishing a round, the currentPlayer integer variable will keep the last player index namely the one from the player who solved the puzzle and thus won the round.

6.Explanation Game.c

In this file, one can found the **loadingBar()** method.

In this method we first set an array of size 27 (because there is one space for the trailing \n, 2 spaces for the [ ] and 24 for all the wedges).Then we add 27 to the random value generated by the method spinWheel(). We designed our loading bar in such way that the loading animation will spin once over the whole array and then, in the second "turn", stops on the wedge corresponding to the generated random number.

In example, if the spinWheel() method generates 17 then the animation in loadingBar() will spin over the 24 wedges + the 17th first wedges of the second wheel turn. That's the reason why we added 27 to the random number (one '[' + 24 wheel wedges + one ']' + one '[')

After this, we finally get to the spinning animation. The general idea is to change a dash into a pipe, return the carriage at the beginning of the line (to overwrite the previously written character) and then print the whole loading bar (with 1 pipe character and 23 dash characters).

At line 18 we use i%25. Like this if random = 41 (17+24), after the first spinning wheel turn when i = 25 (25 not 26 because we starting the for loop with i=1) the considered value is i%25 to return at the beginning of the loading bar array.

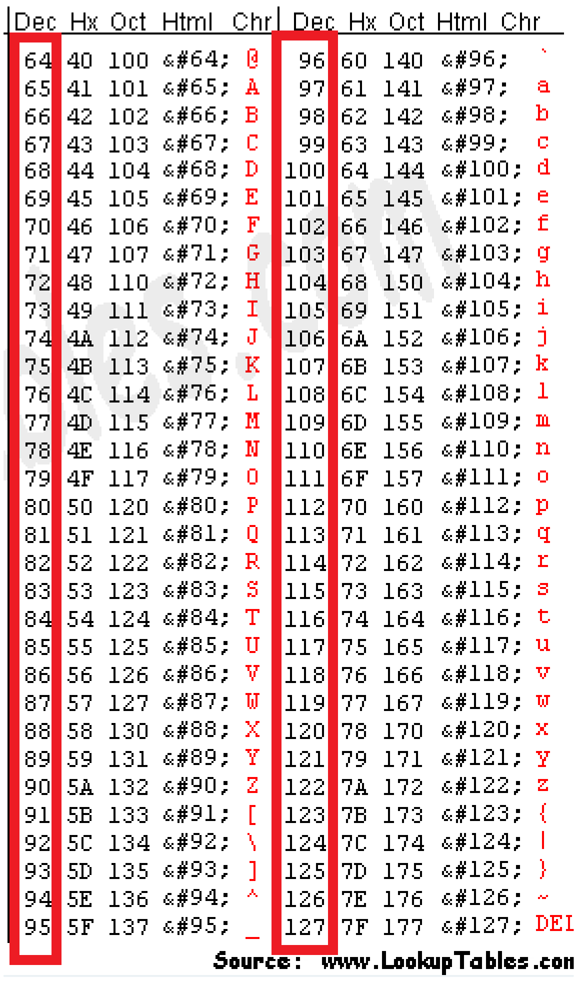
At line 21 we print all the characters. If the character printed is equal to the one changed into a pipe (j = (i%25)) then we print the next character in white ("\033[0m") otherwise we print it in bold red (\033[1;31m). Those codes are referring to ANSI colors.

After this for loop, we paused the program a bit for the animation to be printed correctly (using the sleep() function). Then we set the pipe character back to a dash to give the illusion that the pipe is moving in the array (among the dashes). And finally, if i = 24, meaning the next character is ']' we increment the variable by one so that the next character will be i=1 so the first dash (i=1 and not i=26 because we are using i%25 at line 18).

The method “**getOccurrence**” has the following line:

```
currentPuzzle[i] = c = puzzle[i] > 95 ? tolower(c) : toupper(c);
```

The meaning of this Ternary is: If the integer value of the char is greater than 95 > take the char and lower case it, else upper case it. The reason we chose 95, is because as you can see on the Ascii Table below, any Decimal Values of the character that is above 95 are lower case letters, and below 95 we have the upper case letters.



Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
64	40	100	&#64;	@	96	60	140	&#96;	`
65	41	101	&#65;	A	97	61	141	&#97;	a
66	42	102	&#66;	B	98	62	142	&#98;	b
67	43	103	&#67;	C	99	63	143	&#99;	c
68	44	104	&#68;	D	100	64	144	&#100;	d
69	45	105	&#69;	E	101	65	145	&#101;	e
70	46	106	&#70;	F	102	66	146	&#102;	f
71	47	107	&#71;	G	103	67	147	&#103;	g
72	48	110	&#72;	H	104	68	150	&#104;	h
73	49	111	&#73;	I	105	69	151	&#105;	i
74	4A	112	&#74;	J	106	6A	152	&#106;	j
75	4B	113	&#75;	K	107	6B	153	&#107;	k
76	4C	114	&#76;	L	108	6C	154	&#108;	l
77	4D	115	&#77;	M	109	6D	155	&#109;	m
78	4E	116	&#78;	N	110	6E	156	&#110;	n
79	4F	117	&#79;	O	111	6F	157	&#111;	o
80	50	120	&#80;	P	112	70	160	&#112;	p
81	51	121	&#81;	Q	113	71	161	&#113;	q
82	52	122	&#82;	R	114	72	162	&#114;	r
83	53	123	&#83;	S	115	73	163	&#115;	s
84	54	124	&#84;	T	116	74	164	&#116;	t
85	55	125	&#85;	U	117	75	165	&#117;	u
86	56	126	&#86;	V	118	76	166	&#118;	v
87	57	127	&#87;	W	119	77	167	&#119;	w
88	58	130	&#88;	X	120	78	170	&#120;	x
89	59	131	&#89;	Y	121	79	171	&#121;	y
90	5A	132	&#90;	Z	122	7A	172	&#122;	z
91	5B	133	&#91;	[	123	7B	173	&#123;	{
92	5C	134	&#92;	\	124	7C	174	&#124;	|
93	5D	135	&#93;	]	125	7D	175	&#125;	}
94	5E	136	&#94;	^	126	7E	176	&#126;	~
95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com