CMPUT175-Lab 2 Exercises

The solutions to these exercises have to be shown to your TA in your assigned Lab before the end of your lab time.

To get full marks for this lab, you must complete Exercise 1 and either exercise 2 or 3 (choose one).

1. Write a program that reads in a list of accounts from a file, and then allows the user to enter transactions for each account. The file **accounts.txt** contains a list of accounts with initial balances. As the program runs, it should ask the user for the name of an account, and the amount of money to deposit or withdraw from the account. After each transaction, the program should print the amount remaining in the account. If the **accounts.txt** file is missing, the program should exit with an error message. If any account has a non-float value for the balance, the program should display an error and not add that account. If a non-float is entered for a transaction, or if the account name entered does not exist, the program should print an error message and cancel the transaction. The program should exit when the user enters "**Stop**" for an account name.

Note that the **accounts.txt** file DOES NOT need to be updated by the program.

Input File Example

```
Bob:234.70
Tom Johnson:791.56
Anna:3260.55
Jill:-23.76
Mike Smith:Not_A_Float!
. . .
```

When the line:

Mike Smith:Not_A_Float!

is encountered, the program should print: **Account for Mike Smith not added: illegal value for balance**

If the input file is missing, the program should print: **Input file not found, program will exit**

An example of interaction with the program could look like this:

```
Enter account name, or 'Stop' to exit: Bob
Enter transaction amount: -45.23
New balance for account Bob: 189.47
Enter account name, or 'Stop' to exit: Bob
Enter transaction amount: 117.90
```

```
New balance for account Bob: 307.37
Enter account name, or 'Stop' to exit: Anna
Enter transaction amount: this_is_not_a_float
Illegal value for transaction, transaction canceled
Enter account name, or 'Stop' to exit: Mike Smith
Account does not exist, transaction canceled
Enter account name, or 'Stop' to exit: Stop
```

2 . Write a program that completes the incomplete python class Automobile in the file **Lab2Exercise2.py**, by implementing the methods **get_horsepower()**, **get_color()**, and **get_worth()**. This program should run the **main()** function, which will ask the user for the length, horsepower, and color of an automobile, and will thenprint out the worth of that automobile. An automobile's worth in dollars is calculated as follows:

worth = horsepower*length*color_factor*10

Where color_factor is 3 if the automobile's color is “red”, 2 if the automobile's color is “yellow” or “blue”, and 1 otherwise. For this program, you must use the Automobile class and call the **get_worth()** method. This program does not need to do any exception handling.

```
Example interaction with the program:
Enter automobile's length in meters: 3
Enter automobile's horsepower: 200
Enter automobile's color: white
This automobile is worth $6000.
```

3. Write a number guessing game. The program will start by choosing a random number between 1 and 20 (inclusive). The user has 6 chances to guess what the number is. The program should end when either the user guesses the correct number, or else they have no guesses left. A message should inform the user if they guessed correctly or ran out of guesses. When the user enters a guess, the program must check that the guess is between 1 and 20 (inclusive).

If the guess is not, the program should print a warning message and ask the user to guess again (this should not count as a guess). After each guess, the program should tell the user if the guess is too high or too low. This program does not need to do any exception handling.

Optional – not for marks: Ask the user if they want to play again when they have either made a correct guess or run out of guesses, and then generate a new number and run the game again.

Example interactions with program could look like:

```
Enter a guess (1-20): 10
Too low!
Enter a guess (1-20): 15
Too high!
Enter a guess (1-20): 13
Correct! The number was 13.

Enter a guess (1-20): 14
Too high!
Enter a guess (1-20): 45
That number is not between 1 and 20!
Enter a guess (1-20): 9
Too high!
Enter a guess (1-20): 1
Too low!
Enter a guess (1-20): 4
Too low!
Enter a guess (1-20): 7
Too high!
Enter a guess (1-20): 6
Too high!
You are out of guesses. The number was 5.
```