CMPT 103 – Lab #4

General Information

Python version and IDE:	Python 3.3 / Wing IDE 101
Allocated lab time:	2 hrs and 50 min
Due date:	At the end of the lab period
Lab weight:	2%

Topics

- ✓ Lists
- ✓ Text Processing

Submission

- ✓ **All code file (.py) should be submitted electronically** to your Lab Blackboard site.
- ✓ A portion of the total marks (20%) will be allocated for the programming style. For example, functions should be small; avoid writing duplicate code; names should be meaningful and descriptive; naming conventions should be followed consistently; code should be formatted properly; and comments should be accurate and well written.
- ✓ Comments are **required** for:
  - EACH program indicating the student name and program name.
  - EACH function indicating the function purpose, syntax (example usage of the function), parameters, and return value
  - Any block of code for which the purpose may be unclear (Note: you should always try to write clean code that can be understood easily without comments).

Assignment

For this lab, please put all functions into a file called `Lab4your_initials.py` (e.g., `Lab4FL.py` where F and L are the first letter of your first name and last name). Please feel free to write helper functions if necessary.

Blackjack is a card game involving some players and a dealer. In a hand of blackjack, each player is initially dealt two cards. In real blackjack, players can take additional cards from the dealer. Numbered cards count for their face value, all face cards are worth 10, and an ace is worth 11. The goal of the game is to have the sum of your cards equal to 21. The player closest to 21 wins the round. If multiple players tie for the highest total, they all win.

There are other rules to blackjack, but they're not necessary for this lab. For our game, we will consider the dealer as just one of the players, and players will be dealt exactly two cards each.

1. [10 marks] Create a function called `make_deck` that creates and returns a deck of 52 shuffled cards. The suits Diamonds, Clubs, Hearts, and Spades should be represented by a single character (DCHS). The denominations 2 – 9, Ten, Jack, Queen, King, and Ace should also be represented by a single character (23456789TJQKA). Cards are represented by combining each denomination with each suit. For example, the Queen of Hearts would be denoted "QH".

   Syntax: `deck = make_deck()`
   Parameter: `None`
   Return value: `deck – a list of 52 shuffled cards`

```
>>> deck = make_deck()
>>> print(deck)
    ['4H', '2H', '9S', '3S', '9C', 'QS', '4D', 'KD', 'QC', 'TC', 'JS',
```

Note: the screenshot above does not show all contents of the deck. It is okay to get different results, as `make_deck` should shuffle the cards randomly.

2. [20 marks] Create a function named `deal_blackjack` that deals the cards to some number of players, two cards for each player.

   Syntax: `hands = deal_blackjack(deck, num_of_players)`
   Parameter: `deck – a deck of 52 shuffled cards`
   `num_of_players – the number of players`
   Return value: `hands - a list of lists of cards, one list for each player`

```
>>> hands = deal_blackjack(deck, 5)
>>> print(hands)
    [['4H', '2H'], ['9S', '3S'], ['9C', 'QS'], ['4D', 'KD'], ['QC', 'TC']]
```

3. [15 marks] Write a function named `print_blackjack` that prints out the starting hands of cards. Note that each hand is to be printed vertically.

   Syntax: `print_blackjack(hands)`
   Parameter: `hands – a list containing a list for each hand`
   Return value: `None`

```
>>> print_blackjack(hands)
    4H      9S      9C      4D      QC
    2H      3S      QS      KD      TC
```

4. [25 marks] Write a function named `get_max` that finds the maximum value of all of the hands and returns that maximum value. For full marks on this question, create at least one additional helper function that calculates the value of a single hand and call this function from `get_max`.

   Syntax: `max_value = get_max(hands)`
   Parameter: `hands – a list containing a list for each hand`
   Return value: `the value of the hand with the maximum value`

```
>>> print(hands)
    [['4H', '2H'], ['9S', '3S'], ['9C', 'QS'], ['4D', 'KD'], ['QC', 'TC']]
>>> max_value = get_max(hands)
>>> print(max_value)
    20
```

5. [20 marks] Write a function named `show_winner(hands)` that prints out the hands and marks the winning hand with an asterisk. If more than one hand has the highest total, mark all winning hands. For full marks, this function must call `print_blackjack` and `get_max`.

```
>>> print(hands)
    [['4H', '2H'], ['9S', '3S'], ['9C', 'QS'], ['4D', 'KD'], ['QC', 'TC']]
>>> show_winner(hands)
    4H      9S      9C      4D      QC
    2H      3S      QS      KD      TC
                                    *
```

```
>>> print(hands)
    [['9S', 'QC'], ['5D', '5S'], ['TS', 'KH'], ['2C', '5C'], ['JH', 'TD']]
>>> show_winner(hands)
    9S      5D      TS      2C      JH
    QC      5S      KH      5C      TD
                    *               *
```

6. [10 marks] Write a function that runs the above functions to demonstrate that they work.

Syntax:	`lab_four()`
Parameters:	`None`
Return value:	`None`

This function should perform the following tasks using the functions from above:

- Create and shuffle the deck (using Question 1)
- Deal 5 hands (using Question 2)
- Print out the hands (using Question 3)
- Print out the hands **and** the winner(s) (using Question 5)

```
>>> lab_four()
    Making a deck...
    Dealing 5 hands...

    Printing out the hands...
    JD      2C      5C      5D      7H
    3S      AH      8S      6H      TH

    Showing the winner(s)...
    JD      2C      5C      5D      7H
    3S      AH      8S      6H      TH
                                    *
```