Part 06
Structures & Files

---------- STRUCTURES ----------

1. Fraction calculator

Write a program that allows users to enter 2 fractions and calculate the addition, subtraction, multiplication and division of the 2 entered fractions.

Requirement

- Creates a structure that named Fraction as below:

```
typedef struct Fraction {
	int numerator;
	int denominator;
}
```

- Creates functions includes
  - ✓ `int GCD(int numerator, int denominator)`

    The function `GCD` returns the ***greatest common divisor*** of the numerator and denominator of the fraction. ***Note:*** *denominator can't be zero.*
  - ✓ `struct Fraction input()`

    The function `input` allows user enter value for numerator and denominator of the fraction and return a structure Fraction.
  - ✓ `void output(struct Fraction f)`

    The function `output` uses the GCD function to simplify the fraction `f` and prints out the simplest fraction. ***Note:*** *the fraction must be simplest, for sample:*
    - ✧ *Fraction 0/5 must be displayed as 0*
    - ✧ *Fraction 9/-3 must be displayed as -3*
    - ✧ *Fraction -8/-4 must be displayed as -2*
  - ✓ `struct Fraction add(struct Fraction f1, struct Fraction f2)`

    The function `add` returns the simplest fraction `r = f1 + f2`.
  - ✓ `struct Fraction subtract(struct Fraction f1, struct Fraction f2)`

    The function `add` returns the simplest fraction `r = f1 - f2`.
  - ✓ `struct Fraction multiply(struct Fraction f1, struct Fraction f2)`

    The function `add` returns the simplest fraction `r = f1 * f2`.
  - ✓ `struct Fraction divide(struct Fraction f1, struct Fraction f2)`

    The function `add` returns the simplest fraction `r = f1 / f2`.

- ✓ `int menu()`

  The function **`menu`** returns the selection of user after displayed the function list:

  1. Adds two fractions
  2. Subtracts two fractions
  3. Multiplies two fractions
  4. Divides two fractions
  5. Quit

- ✓ `int main()`

  The function **`main`** uses **`menu`** function to display menu and allows user to calculate addition, subtraction, multiplication and division based on built functions.

```
   *** THE FRACTION CALCULATOR ***
   ================================
1  Adds two fractions
2  Subtracts two fractions
3  Multiples two fractions
4  Divides two fractions
5  Quit
    Please select function (1-5): 10
    >> The function must be from 1 to 5! Please select again: 1

    +++ ADDS TWO FRACTION +++
    ------------------------
Please the first  fraction: 4/0
>> The denominator can't be zero! Please enter again: 4/5
Please the second fraction: 2/8
The result is 4/5 + 2/8 = 21/20

   *** THE FRACTION CALCULATOR ***
   ================================
1  Adds two fractions
2  Subtracts two fractions
3  Multiples two fractions
4  Divides two fractions
5  Quit
    Please select function (1-5): 2

    +++ SUBTRACTS TWO FRACTION +++
    -----------------------------
Please the first  fraction: -8/5
Please the second fraction: 7/-2
The result is -8/5 - 7/-2 = 19/10

   *** THE FRACTION CALCULATOR ***
   ================================
1  Adds two fractions
2  Subtracts two fractions
3  Multiples two fractions
4  Divides two fractions
5  Quit
    Please select function (1-5): 3

    +++ MULTIPLES TWO FRACTION +++
    -----------------------------
Please the first  fraction: 9/4
Please the second fraction: 8/3
The result is 9/4 * 8/3 = 6

   *** THE FRACTION CALCULATOR ***
   ================================
1  Adds two fractions
2  Subtracts two fractions
3  Multiples two fractions
4  Divides two fractions
5  Quit
    Please select function (1-5): 4

    +++ DIVIDES TWO FRACTION +++
    ---------------------------
Please the first  fraction: -4/10
Please the second fraction: 9/5
The result is -4/10 / 9/5 = -2/9

   *** THE FRACTION CALCULATOR ***
   ================================
1  Adds two fractions
2  Subtracts two fractions
3  Multiples two fractions
4  Divides two fractions
5  Quit
    Please select function (1-5): 5

## Thank for using our software ##
## Goodbye!                     ##
## See you again!               ##
```

---------- TEXT FILES ----------

2. File Handling

Background Context

A file manager or file browser is a computer program that provides a user interface to manage files and folders. The most common operations performed on files or groups of files include creating, opening (e.g. viewing, playing, editing or printing), renaming, moving or copying, deleting and searching for files, as well as modifying file attributes, properties and file permissions. Folders and files may be displayed in a hierarchical tree based on their directory structure. Some file managers contain features inspired by web browsers, including forward and back navigational buttons.

Program Specifications

File stores information for many purposes and retrieve whenever required by our programs. A file represents a sequence of bytes on the disk where a group of related data is stored. File is created for permanent storage of data. C programming language can handle files as Stream-oriented data (Text) files and System oriented data Binary files.

Function details:

- ✓ **Function 1: Read file**

  This feature reads a file entered by the user and displays its contents on the screen. Opening a file means we bring file from disk to ram to perform operations on it. The file must be present in the directory in which the executable file of this code sis present.

- ✓ **Function 2: Copy files**

  Firstly you will specify the file to copy and then you will enter the name of target file, You will have to mention the extension of file also. We will open the file that we wish to copy in read mode and target file in write mode.

- ✓ **Function 3: Merge two files**

  Merges two files and stores their contents in another file. The files which are to be merged are opened in read mode and the file which contains content of both the files is opened in write mode. To merge two files first we open a file and read it character by character and store the read contents in another file then we read the contents of another file and store it in file, we read two files until EOF (end of file) is reached.

- ✓ **Function 4: List files in a directory**

  List all files present in a directory/folder in which this executable file is present.

- ✓ **Function 5: Delete file**

  Deletes a file which is entered by the user, the file to be deleted should be present in the directory in which the executable file of this program is present. Extension of the file should also be entered, remove macro is used to delete the file. If there is an error in deleting the file then an error will be displayed using perror function.

Expectation of User interface

```
1) Read file
2) Copy files
3) Merge two files
4) List files in a directory
5) Delete file

Choice features: 4

a.out
a.txt
b.txt
pgm.c
pgm1.c
pgm10.c
pgm11.c

Choice features: 1

Enter the name of file you wish to see: a.txt

Hello world!!

Choice features: 2
Enter name of file to copy: a.txt
Enter name of target file: c.txt
File copied successfully!!

Choice features: 3
Enter name of first file: a.txt
Enter name of second file: b.txt
Enter name of file which will store contents of two files: result.txt
Two files were merged into result.txt file successfully!!

Choice features: 5
Enter the name of file you wish to delete: result.txt
result.txt file deleted successfully!!!
```

3. Simple Slot Machine

Background Context

This is a simple slot machine that does not resemble the real machines in a casino. However, it does demonstrate the concept of probability in an actual slot machine.

Program Specifications

Write a program that allows the user to play a simplified version of a slot machine. For the registered user, their budget will be loaded from the data text file "simpleSlotMachine.txt". Since the budget of the user is empty or it's the first time the user plays the game, they will start with $10. At the beginning of the program the user is given the following choices:

1. Play the slot machine.
2. Save game
3. Cash out.

Each play costs 25 cents. For each play, the slot machine will output a random three-digit number, where each digit ranges from 0 to 9. (This can be done by generating three separate integers in between 0 and 9, inclusive.) The result of playing the slot machine should be outputted along with what the user won. Here is a chart to indicate the winners earnings:

Combination	Winnings
2 of a kind (such as 676 or 112)	50 cents
3 of a kind (000, 111, ..., 999)	$10.00

Thus technically, you make a net earnings of 25 cents if you get a 2 of a kind and a net earnings of $9.75 when you get 3 of a kind. If a player runs out of money, automatically cash them out and tell them that they have no money left. Thus, a player, after losing who has 0 cents left should not be prompted with the menu at all. Rather, they should simply be presented with a message to end the game. Otherwise, when a player chooses to cash out, thank them for playing and output how much money they have left.

Function details

1. The choice entered by the user for the menu will always be valid.
2. If you choice 2, save your money to file. And restore the money in the play next time .
3. If you have no money to play any to more, say good bye.

Expectation of User interface

```
You have $10.00.
Choose one of the following menu options:
1) Play the slot machine.
2) Save game.
3) Cash out.
1
The slot machine shows 046.
Sorry you don't win anything.
You have $9.75.
Choose one of the following menu options:
1) Play the slot machine.
2) Save game.
3) Cash out.
1
The slot machine shows 933.
You win 50 cents!
You have $10.00.
```

```
Choose one of the following menu options:
1) Play the slot machine.
2) Save game.
3) Cash out.
1
The slot machine shows 444.
You win the big prize, $10.00!
You have $19.75.
Choose one of the following menu options:
1) Play the slot machine.
2) Save game.
3) Cash out.
2
Your money had saved!
1) Play the slot machine.
2) Save game.
3) Cash out.
3
Thank you for playing! You end with $19.75!
```

4. Hangman

Background Context

An airline reservation system (ARS) is part of the so-called passenger service systems (PSS), which are applications supporting the direct contact with the passenger.

ARS eventually evolved into the computer reservations system (CRS). A computer reservation system is used for the reservations of a particular airline and interfaces with a global distribution system (GDS) which supports travel agencies and other distribution channels in making reservations for most major airlines in a single system.

Program Specifications

You are given a puzzle with blanks (representing letters) and you guess a letter that might be in the puzzle. If it is, then each occurrence of the letter is placed on the board. If not, you get a point against you (which is usually drawing one portion of a stick-figure.) If you complete the puzzle before too many wrong guesses (this usually depends on how many parts there are to the stick-figure being drawn), then you win. If not, you lose. For our game, we won't draw any stick figures, but we'll simply keep track of how many incorrect guesses the user has made. In order to win, the user must uncover the phrase before they get 5 incorrect guesses.

For this game, all of the words to be guessed will have uppercase letters only. All of the possible puzzles will be read in from a file into a two-dimensional character array (array of strings). The code for this will be given to you in a separate file. You will have to call the given function only once in your program. From there, each time the user wants to play the game, you'll pick a different random word stored in the array of strings as the word for the puzzle.

Note: *All words are guaranteed to be 29 characters or fewer and the file is guaranteed to have 1000 words or fewer...*

Function details

At the beginning of the program, you should prompt the user to enter the name of the input file with all of the words for the game.

Then, you should ask the user if they want to play. If they do, choose a random puzzle, and present it to them with all the letters hidden represented by underscores and ask them what letter they want to choose. Separate each slot with a blank. Here is an example:

```
You currently have 0 incorrect guesses.
Here is your puzzle:

_ _ _ _
Please enter your guess.
```

After the user enters their guess, if the letter they guessed is in the puzzle and hasn't been guessed before, print out the following message:

```
Congratulations, you guessed a letter in the puzzle!
```

If the letter is NOT in the puzzle, print out the following message:

```
Sorry, that letter is NOT in the puzzle.
```

If the user has ALREADY guessed that letter, print out the following message:

```
Sorry, you have guessed that letter already.
Now it counts as a miss.
```

Once you've printed this out, continue to the next turn, prompting the user with the board again as described above.

If the user has 5 incorrect guesses, instead of re-prompting them with the board, print out the following message:

```
Sorry, you have made 5 incorrect guesses, you lose.
The correct word was ____.
```

Where ____ is the word for the round.

If the user fills in the entire word with the letters they guess before 5 incorrect guesses, print out the following message:

Congratulation! You got the correct word, ____.

Where ____ is the word for the round.

After the round is over, prompt the user if they want to play again. If so, repeat the process described here.

Expectation of User interface

```
What file stores the puzzle words?
words.txt

Would you like to play hangman (yes,no)?
yes

You currently have 0 incorrect guesses.
Here is your puzzle:
_ _ _ _

Please enter your guess.
A

Congratulations, you guessed a letter in the puzzle!

You currently have 0 incorrect guesses.
Here is your puzzle:
_ A _ _

Please enter your guess.
B

Sorry, that letter is NOT in the puzzle.

You currently have 1 incorrect guesses.
Here is your puzzle:
_ A _ _

Please enter your guess.
M

Congratulations, you guessed a letter in the puzzle!

You currently have 1 incorrect guesses.
Here is your puzzle:
M A _ _

Please enter your guess.
A

Sorry, you have guessed that letter already.
Now it counts as a miss.

You currently have 2 incorrect guesses.
Here is your puzzle:
M A _ _

Please enter your guess.
L

Congratuations! You got the correct word, MALL.

Would you like to play hangman (yes,no)?
yes

You currently have 0 incorrect guesses.
Here is your puzzle:
_ _ _ _ _ _ _ _

Please enter your guess.
E

Sorry, that letter is NOT in the puzzle.
```

```
You currently have 1 incorrect guesses.
Here is your puzzle:
_ _ _ _ _ _ _ _

Please enter your guess.
A

Congratulations, you guessed a letter in the puzzle!

You currently have 1 incorrect guesses.
Here is your puzzle:
_ _ _ _ _ A _ _

Please enter your guess.
R

Sorry, that letter is NOT in the puzzle.

You currently have 2 incorrect guesses.
Here is your puzzle:
_ _ _ _ _ A _ _

Please enter your guess.
S

Sorry, that letter is NOT in the puzzle.

You currently have 3 incorrect guesses.
Here is your puzzle:
_ _ _ _ _ A _ _

Please enter your guess.
N

Sorry, that letter is NOT in the puzzle.

You currently have 4 incorrect guesses.
Here is your puzzle:
_ _ _ _ _ A _ _

Please enter your guess.
C

Sorry, you have made 5 incorrect guesses, you lose.
The correct word was FOOTBALL.

Would you like to play hangman (yes,no)?
no

Thanks for playing!
```

Guidelines

Store two strings for the program: one which is the correct word, and the other which is what gets printed out to the user. For example, if the correct word was MALL, store the two following strings:

```
MALL
____
```

When the user guesses a letter, read it into a string and just access index 0 of that string to find the character they guessed. (This will make dealing with white space easier.)

Scan through the real word, checking to see if any of the letters equals the guessed letter. If so, replace the corresponding character (at the same index) in the other string. So, for example if A were guessed, the two strings would then look like:

```
MALL
_A__
```

To keep track of which letters have been guessed and which ones haven't, store an integer array of size 26 and initialize it to all 0s, indicating that no letter had been guessed. When a letter is chosen, go to the appropriate index and add 1 to the value stored there. For example, if the letter chosen is stored in guess[0], and the name of the array is chosen, we would do the following:

```
chosen[(int)(guess[0] - 'A')]++;
```

Since the ASCII values are contiguous for capital letters, guess[0] – 'A' will equal a value from 0 to 25. It will equal the index which stores whether or not that letter has been guessed.

To detect if the puzzle has been guessed, just check if the two separate strings are equal.

To detect if the user has lost, just keep a single integer variable that stores the number of incorrect guesses and check if that equals 5 (which should be stored in a constant.)

To print out the word for the user, write a function which prints out one letter from the string followed by a space, repeated throughout the word, so that

```
_ A _ _
```

gets printed instead of

```
_A__
```

5. Casino player

Background Context

To give students practice in writing functions and calling those functions to perform more complex tasks.

Program Specifications

You will write a program that simulates a casino for a single player. The user will initially start with $1000. The user will then be able to choose from the following options:

1. Buy chips
2. Sell chips
3. Play Craps
4. Play Arup's Game of Dice
5. Status Report
6. Quit

Your program will execute each choice until the quits. At this point all of their chips automatically get sold back to the casino and a message prints out how much money the user has left (of the $1000) after gambling.

Function details

✓ One of the most "fair" games to play at a casino is Craps. Here is one version of how to play:

1. Roll a pair of fair six-sided dice.
2. If you roll a 7 or 11, you win!
3. If you roll a 2, 3, or 12, you lose.
4. Otherwise, record what you've rolled. Let this sum be k; also known as your point.
5. If you rolled a point, continue rolling the pair of dice until you get either your point (k) or a sum of seven on the two dice.
6. If k comes up first, you win!
7. If 7 comes up first, you lose.

- ✓ Amazingly, this game is even more "fair" than Craps, but the house still has a 50.2% chance of winning, which is why the casino hasn't gone broke yet! Here are the rules:
  1. Roll a pair of dice.
  2. If you roll a sum of 11 or 12, you win.
  3. If you roll a sum of 2, you lose.
  4. Otherwise, record what you've rolled. Let this sum be k; also known as your point.
  5. Roll one more time. If this roll exceeds your point(k), you win!
  6. If this roll is the same as your point(k) or lower, you lose.
- ✓ Chips cost $11. Whenever a customer buys chips, he/she must give the banker some money. The banker will always give the user the maximum number of chips they can buy with the money given to them and return the leftover cash. You will write a single function that takes care of this transaction.
- ✓ The casino buys chips back at $10 a piece. You will write a single function that takes care of this transaction.

Expectation of User interface

```
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
1
How much cash do you want to spend for chips?
1100
Sorry, you do not have that much money. No chips bought.
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
1
How much cash do you want to spend for chips?
500
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
5
You currently have $505 left and 45 chips.
```

```
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
3
How many chips would you like to bet?
0
Sorry, that is not allowed. No game played.
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
3
How many chips would you like to bet?
10
Press 'r' and hit enter for your first roll.
r
You rolled a 8.
Press 'r' and hit enter for your next roll.
```

```
r
You rolled a 12.
Press 'r' and hit enter for your next roll.
r
You rolled a 6.
Press 'r' and hit enter for your next roll.
r
You rolled a 7.
Sorry, you have lost.
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
4
How many chips would you like to bet?
17
Press 'r' and hit enter for your first roll.
r
You rolled a 3.
Press 'r' and hit enter for your next roll.
r
You rolled a 8.
You win!
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
6
After selling your chips, you have $1025. Thanks for playing!

Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
1
How much cash do you want to spend for chips?
500
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
2
How many chips do you want to sell?
46
Sorry, you do not have that many chips. No chips sold.
```

```
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
4
How many chips would you like to bet?
46
Sorry, you do not have that many chips to bet. No game played.
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
4
How many chips would you like to bet?
5
Press 'r' and hit enter for your first roll.
r
You rolled a 11.
You win!
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
4
How many chips would you like to bet?
10
Press 'r' and hit enter for your first roll.
r
You rolled a 5.
Press 'r' and hit enter for your first roll.
r
You rolled a 5.
Sorry, you have lost.
Welcome to the Casino. Here are your choices:
1) Buy chips
2) Sell chips
3) Play Craps
4) Play Arup's Game of Dice
5) Status Report
6) Quit
6
After selling your chips, you have $905. Thanks for playing!
```

7. Spell Checker

Background Context

Many of us have horrible spelling and would get great practical use out of a spell-checker. In this assignment, you will write a simplified version of a spell checker. In the process, you will write several of your own string functions.

Program Specifications

The first line of the file will have a single positive integer, n ($n \leq 30000$), representing the number of words in the file. The following n lines will contain one word each, in alphabetic order. All words will contain lowercase letters only and will be no longer than 29 letters long.

Function details

4. Your program should first read in the dictionary into a two-dimensional character array (from dictionary.txt) and print a message to the screen stating that the dictionary has been loaded.
5. Then your program should prompt the user to enter a word, all lowercase.
6. If the word is in the dictionary, you should simply print out a message stating this.
7. If the word is NOT in the dictionary, then you should provide a list of possible suggestions (of correctly spelled words that the user might have been trying to spell) in alphabetical order.
8. Here are the criteria for whether or not a real word should end up on this list:
   - If either string is a sub string of the other, and the lengths of the two strings are within 2 of one another.
   - If either string is a sub sequence of the other and the lengths of the two strings are within 2 of one another. (This would get rid of the "car", "camera" case, for example.)
   - If the strings are permutations of one the other. (Only try this test if the two strings are of the same length.)
   - If the match score between the two strings is less than 3. (Only try this test if the two strings are of the same length.)
9. Continue prompting the user to enter words until they decide that they don't want to enter any more. (Thus, they will be forced to enter the first word, but after that, you'll provide them a choice as to whether or not they want to enter another word to check

Expectation of User interface

```
Welcome to the Spell Checker!
The dictionary has been loaded.
Please enter the word you would like checked.
flag
Great, flag is in the dictionary!
Would you like to enter another word? (yes/no
yes
Please enter the word you would like checked:
peice
Here are the possible words you could have me
alice
beige
brice
deuce
fence
heine
hence
ice
juice
paine
peace
peach
peale
pease
pee
place
poise
ponce
price
prick
pride
prime
prize
reich
seize
slice
spice
twice
voice
Would you like to enter another word? (yes/no)
yes
Please enter the word you would like checked.
independant
Here are the possible words you could have meant:
independent
Would you like to enter another word? (yes/no)
no
```

Guidelines

Note: *Assume that all of the formal parameters for each of the functions below are lowercase alphabetic strings.*

A string is a sub string of another one if all of its letters appear contiguously (in the same order) in the second string. For example, "bound" is a sub string of "home bound" and "ire" is a sub string of "firefly", but "car" is NOT a sub string of "camera" because the letters 'm' and 'e' are in between the letters 'a' and 'r' in "camera."

```
// Returns true if shortstr is a sub string of longstr, and false otherwise.
int substring(char shortstr[], char longstr[]);
```

A string is a sub sequence of another string if all the letters in the first string appear in the second string, in the same ordering, but not necessarily contiguously. For example, "car" is a sub sequence of "camera", since the letters 'c','a', and 'r' appear in that order (but not contiguously) in "camera." Similarly, "hikes" is a sub sequence of "chickens," but "tale" is NOT a sub sequence of "wallet" because the letter 't' occurs AFTER the letter 'a' in "wallet."

```
// Returns true if shortstr is a sub sequence of longstr, and false otherwise.
int subsequence(char shortstr[], char longstr[]);
```

A permutation of letters is simply a different ordering of those letters. For example, "care" is a permutation of "race" and "spill" is a permutation of "pills," but "apple" is NOT a permutation of "pale" because the letter 'p' appears only once instead of twice in "pale." A natural consequence of this definition is that two strings can only be permutations of one another if they are both the same length.

```
// Returns true if string1 and string2 are permutations
// of each other, false otherwise.
int permutation(char string1[], char string2[]);
```

When we are comparing two strings of equal length, we can define a term called "match score" which is simply the number of corresponding letters in which the two strings disagree. For example, the match score between "drink" and "blank" is 3 because the first three letters don't match. The match score between "marker" and "master" is two because letters 3 (r vs. s) and 4 (k vs. t) do not match.

```
// Precondition  : string1 and string2 are the same length
//                 and only contain lowercase letters.
// Post-condition: returns the score of the match score between these two strings.
int matchscore(char string1[], char string2[]);
```

---------- BINARY FILES ----------

8. English-English Dictionary

Program Specifications

Create an English – English dictionary program.

Main functions as below:

1. Create a new word
2. Edit a word
3. Look up meaning
4. Exit

Function details

1. Create a new word: allow user create a new word and its meaning
2. Update an existing word
3. Look up meaning

Expectation of User interface

```
1. Create a new word
2. Edit a word
3. Look up meaning
4. Exit
Please choose number (1 – 4):
```

- **User choose 1:**

```
Enter a new word: Go [geu]
Meaning: Move from one place to another
```

- **User choose 2:**

```
Enter a word to update: Go
Meaning: Move from one place to another; Travel
```

- **User choose 3:**

```
Enter a word to look up: Go
Meaning: Move from one place to another; Travel
```

Guidelines

Data should be stored in text file, keyword is stored in an index file, word meaning is stored in data file.

Data should be stored in each category, for example: all words start with ‘a’ stored in a_index.dat and a_meaning.dat files

9. Manage Student

Program Specifications

Create a program to manage students, student information includes:

- ✓ Student code
- ✓ Student name
- ✓ Date of birth
- ✓ Learning point

Student information is input from keyboard and stored in a text file name "Student.txt". The program allows user enter new students, look up student and display student list.

Function details:

✓ **Main menu for program as below:**

1. Enter student list
2. Look up student
3. Display student list
4. Exit

The program will implement corresponding function base on the number user input, user press 1 then the program implement function for menu "Enter student list", press 2 then the program run "Look up student", press 3 the program run "Display student list", press 4 then exit.

Menu must be display the first when running the program.

✓ **Enter student list:**

This function is implement when choose menu "Enter student list", student information is stored in a text file name "Student.txt". Student list in the text file must be sorted in student name ascending order.

✓ **Look up student**

This function is implement when choose menu "Look up student", look up student base on student name input from keyboard, display found student information on the screen

✓ **Display student list**

This function is implement when choose menu "Display student list", all students from the file will be display on the screen.

Expectation of User interface:

Menu of the program display the first must be as below:

```
1. Enter student list
2. Look up student
3. Display student list
4. Exit
      Please choose menu (1 – 4):
```

- **When user press 1 the program display:**

```
Enter new student:
Student code: SV001
Student name: Ana
Date of birth: 11-Jan-1991
Learning point: 3.7
      >> Press enter to continue, ESC to return the main menu
```

- **When user press 2:**

```
Please enter student name: John
If found then display student information name John:
Student code: SV001
Student name: John Mark
Date of birth: 12-Jan-1990
Learning point: 4.3
      >> Press enter to continue, ESC to return the main menu
```

- **When user press 3:**

```
Student list:
--------------------------
Student code: SV001
Student name: Jack
Date of birth: 11-Nov-1994
Learning point: 3.5
-----------------
Student code: SV002
Student name: Daisy
Date of birth: 02-May-1994
Learning point: 3.6
------------------------
      >> Press enter to continue, ESC to return the main menu
```

- **When user press 4:**

```
exit the program
```

Guidelines

When store student information in the text file should be setup mark-point to separate data between students, for example: Mark-point is '#'.

10. Fruit Shop

Background Context

Fruit Shop management system in C is basically developed for manage the Fruit Shop. In the Fruit Shop, product and Shopping management is very important. By making system is computerized it make possible to reduce effort, work is efficient and increase their revenue opportunities for shop owner.

Program Specifications

The program provides shop owners tools to run their business effectively. The program's functions as below:

```
    FRUIT SHOP SYSTEM
1. Create Fruit
2. View orders
3. Shopping (for buyer)
4. Exit
    Please choose:
```

Function details:

- ✓ **For Fruit Shop owner**
  - ✧ ***Create product (Fruit):***
    - ◆ A Fruit has attributes: Fruit Id, Fruit Name, Price, Quantity and Origin.
    - ◆ From "Main Screen", use select item (1) to create Fruit. After each Fruit is created, the system shows message: `Do you want to order now (Y/N)` ? User chooses Y to continues, if you chooses N, the program returns main screen and display all Fruits what are created.

- ✧ ***View orders***
  - ◆ To view orders list, who buy and how many product

```
Customer: Marry Carie
+-----+--------------------+----------+-------+--------+
| No. | Product            | Quantity | Price | Amount |
+-----+--------------------+----------+-------+--------+
|   1 | Apple              |        3 |    1$ |     3$ |
|   2 | Mango              |        2 |    2$ |     4$ |
+-----+--------------------+----------+-------+--------+
|                                    TOTAL |     7$ |
+-----+--------------------+----------+-------+--------+

Customer: John Smith
+-----+--------------------+----------+-------+--------+
| No. | Product            | Quantity | Price | Amount |
+-----+--------------------+----------+-------+--------+
|   1 | Jack Fruit         |        3 |    3$ |     9$ |
|   2 | Mango              |        2 |    2$ |     4$ |
+-----+--------------------+----------+-------+--------+
|                                    TOTAL |    13$ |
+-----+--------------------+----------+-------+--------+
```

- ✓ **Shopping**
  - ✧ Customer selects item 3, the program displays all fruits. For example:

```
List of Fruit:
+-----+--------------------+-------------+-------+
| No. | Fruit Name         | Origin      | Price |
+-----+--------------------+-------------+-------+
|   1 | Coconut            | Vietnam     |    2$ |
|   2 | Orange             | US          |    3$ |
|   3 | Apple              | Thailand    |    4$ |
|   4 | Grape              | France      |    6$ |
|   5 | Mango              | Vietnam     |    2$ |
+-----+--------------------+-------------+-------+
```

  - ✧ To order, customer selects Item, since customer selects item 1, the program shows:

```
You selected: Coconut
Please input quantity:
```

  - ✧ After customer inputs quantity of fruit, the program shows message: `Do you want to order now (Y/N)`. If customer selects N, the program returns to List of Fruit to continue ordering. If select Y, the program displays:

```
+--------------------+----------+-------+--------+
| Product            | Quantity | Price | Amount |
+--------------------+----------+-------+--------+
| Coconut            |        3 |    2$ |     6$ |
+--------------------+----------+-------+--------+
Input your name:
```

  - ✧ Customer inputs his/her name to finish ordering. The program returns main screen.