Unit 2: Sequence, Selection, Iteration

Classwork (4 Tasks)

N.B Look at Appendix B for a summary of Selection and Iteration concepts

3.1 Game Store

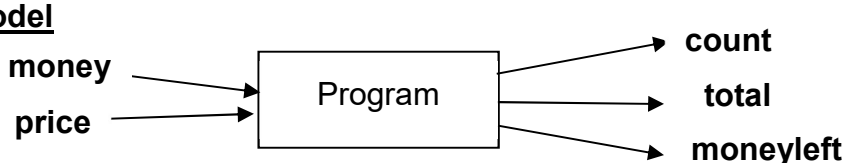
For this program, the design has been done for you. You are to write the code, and then compile, run and test the program using the test plan.

Problem

At the start of a game, you enter a Game Store where you are allowed to buy 6 weapons to help you later in the game. Design and write a program that inputs the amount of money you have to spend. The program should then input the price of each of the weapons bought and keep a total of the amount spent. At the end of the program, output how many items were bought, the total amount spent and the money left (if any).

Program Design

Input-Output Model



Identifier List

Identifier	Data Type	Meaning
money	double	*money to spend*
price	double	*price of one weapon*
count	int	*count of weapons bought*
total	double	*total cost of weapons bought*
moneyleft	double	*amount left after purchases*

Algorithm

1. Prompt the player and input the amount of **money** to spend
2. Output instructions to enter the prices of items when prompted
3. Initialise variables
   total = 0
   count = 0
4. Loop while (**count < 6**)
   a. add **1** to **count**
   b. Output a message to enter price of item number: **count**
   c. Input **price**
   d. Add **price** to **total**
   e. Calculate **moneyleft**

   End Loop
5. Output **count** of items and **total** cost of items
6. Output **moneyleft**

Test plan

Test No.	Inputs		Expected Outputs			Actual Outputs		
	Money	price	count	total	moneyleft	count	total	moneyleft
1	100	10, 5, 4, 2, 20, 3	6	44.00	56.00			

2	80	6.75, 18.5, 0.36, 12.6, 1, 24	6	63.21	16.79			
3	200	0, 0, 0, 0, 0, 0	6	0.00	200.00			
4	200	50, 30, 40, 60, 40, 30	6	180	20.00			

Task 3.1

1. You are to code the program. You must follow the design given on the previous page.
2. Then test the program using the test plan above.

Extra: Notice that the program allows you to purchase weapons even when you have run out of money!! (**see Test 4 above**).

- Modify the program so it stops either when money runs out or the count reaches 6.
- Test the program thoroughly to check that it now works properly

Put source code, completed test plan and screenshots into your logbook

3.2 Loops and Screen Displays

Task A (using a while loop)

Write a new program that does the following (there are some hints on the next page)

- new colours are set for the screen and text display (see Appendix A).
- the user is asked to enter their name
- a **while loop** is used to print this name on the screen **10 times**
  - Each name is printed underneath the previous one
  - There is a pause of 0.5 seconds between each name (see **Hint 1** on next page)

Task B (using another while loop)

Now add some more code after Task A to do the following:

- the screen is cleared using different colours
- the name that was previously entered moves down the screen (one name .. not 10)
- Here is a partial algorithm which may help you:

1. Set x and y values for print position
2. **Loop** while count < 10
   - Set cursor position using x and y (see Hint 2)
   - Print the name
   - Add 1 to y value (to move down screen)
   - Add 1 to count (to keep loop going)
   - Pause for 0.5 seconds
   - Clear screen to remove previous name

End Loop

Task C (using a for loop)

Now add some more code underneath the previous program code to print the user name at random positions on the screen.
This partial algorithm should help:

1. Set new (different) screen and text colours
2. Clear the screen
3. Use a **for loop** to repeat the following 10 times:
    - Pick a random value for x between 0 and 80 (see Hint 3)
    - Pick a random value for y between 0 and 24
    - Set cursor position using x and y
    - Print the name
    - Pause for 0.5 seconds

   End Loop

Hints

1. A **Pause** can be achieved by using:
   System.Threading.Thread.Sleep(1000); *// this pauses program for 1 second*
2. **Cursor Position** can be set using:
   Console.SetCursorPosition(x, y); *// the next output will be at position x,* y
3. **Random Numbers** can be generated like this:
   Random rand = new Random(); *// do this once at start of program*
   x = rand.next(10); *// x = a random number from 0 to 10*
4. **3 Programs in ONE**
   Console.Readkey(); *// use this to pause the program between sections*

Put all source code and sample screenshots in your logbook

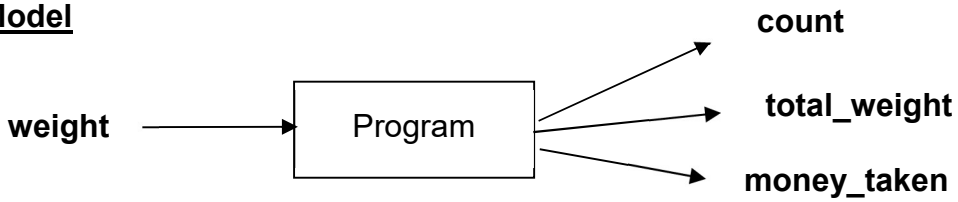
3.3 Space Ferry

In one part of a game, a small **Space Ferry** is used to cross dangerous territory.
We have to design a program for 'Sharko's Space Ferry Service'. Sharko waits until he has a full load before taking off, but his small rocket can only handle a maximum load of **2400kg**.

a. Our program will input the weight of each passenger (in kg);
b. The program will keep a passenger count and also a total for the passenger weight which must not exceed the maximum weight limit.
c. Any remaining passengers are told to wait for the next rocket.
d. At the end of the program the program outputs the passenger count and the total passenger weight
e. Each passenger is charged 60 credits .. the total collected is displayed at the end

Here is the design for the program ..

Input-Output Model



Identifier List

Identifier	Data type	Meaning
weight	Double	*Weight of a passenger*
total_weight	Double	*cumulative passenger weight*
count	Int	*count of passengers*
money_taken	double	*money paid by passengers*
Constants	**Value**	**Meaning**
MAX_WT	**=2400**	*Maximum passenger weight*

Algorithm

1. Output a suitable heading for Sharko's Space Ferry
2. Initialise
total_weight = 0
count = 0
3. Output a message to enter a passenger weight in kgs
Input **weight**
4. Loop while **total_weight** + **weight** <= **MAX_WT**
a. add **1** to **count**
b. add **weight** to **total_weight**
c. Output a message saying passenger accepted aboard
d. Output a message to enter a passenger weight in kgs
e. Input **weight**
End Loop
5. Output a message apologizing that No more passengers can board the ferry.
6. Output **total_weight** and **count**
7. Calculate and output the **money_taken**
8. Output a final Lift-Off message

Study this condition. It is designed to only accept a passenger if their **weight** added to **total_weight** is less than or equal to **MAX_WT**

Note: This algorithm uses the 'read-ahead method'

Test plan

	TEST 1			TEST 2		
	Input	**Outcome**	**Result**	**Input**	**Outcome**	**Result**
weight	320	passenger accepted		430	passenger accepted	
weight	400	passenger accepted		390	passenger accepted	
weight	450	passenger accepted		400	passenger accepted	
weight	280	passenger accepted		410	passenger accepted	
weight	310	passenger accepted		380	passenger accepted	
weight	285	passenger accepted		390	passenger accepted	
weight	290	passenger accepted		280	***passenger rejected***	
weight	300	***passenger rejected***				
	Expected Output	**Actual Output**		**Expected Output**	**Actual Output**	
total_ weight	2335			2400		
count	7			6		
money _taken	420			360		

Task 3.4

1. You are to write the program .. following the design given to you.
   - Compile, run and test the program using the above test data.
2. Add more code to produce a warning message when the total weight gets within 200 kg of the maximum allowed.
   - Test this again to check that it works
3. Put the source code, sample screenshots into your logbook

3.4 High Pressure

A chemical company needs a program to check **pressures** in a tank of dangerous chemicals and to output various warning messages when the pressure reaches certain values.

- Your program must continually input the **pressure** using a repeating loop.
  (Hint: you may find the read-ahead method used previously is a good approach)
- It is **safe** as long as the pressure is less than **150** units, but if the pressure reaches 150 or more, the chemical plant must shut down immediately.
- One of these warning messages is to be output, depending on the circumstances :
  i. **Normal** operating pressure (10 to 100 units) .
  ii. Under 10 units is **Too Low** operating pressure.
  iii. Over 100 units is **Too High**
  iv. Over 125 is **Dangerous**.
- Output an appropriate message for each situation and also a shut-down message if the plant is closed down (the loop should then finish).
- Put the source code and sample outputs into your logbook

Independent Study (2 Tasks)

The following exercises are to be done individually and independently, in your own time.

3.5 Game ON

Use a do-while loop in the following program.

Design, write and test a simple **guessing game** between a player and the computer:

a) Input the name of the player and give him/her some instructions.
b) The computer picks a secret random number between 1 and 100 .. which is not displayed (See the note below on how to select a random number).
c) Input the player's guess for the secret number.
d) Compare the player's guess with the secret number and display a message telling the player whether they are 'Too High', 'Too Low' or 'Spot ON!'
e) This guessing process (steps c and d) repeats until the correct number is guessed.
f) The player is then told how many guesses they took, and also how well they did (e.g. more than 10 guesses: terrible!, less than 5: very good, etc.)
g) Ask if another player wants a go and input a reply of 'y' or 'n'.
h) Repeat the whole program while the reply is not 'n'.
i) Output the number of players that have played the game at the end.

Put source code and sample screenshots into your logbook

How to select a random number

- C# has a built-in Class called **Random** that we can use for this.
- First we must create a new object from this class ..
  e.g.

```
Random  rand = new  Random();      // creates a new object called rand
```

- Then you can use **rand** to pick the next number e.g.

```
    // pick a random number between 1 and 6 and store in an int variable n
n = rand.Next(6)  + 1;   // picks a random number between 1 and 6
```

See Next Page for a typical game session:

A Typical Game Session

```
Brian's Guessing Game
==================
Enter Your Name:    Joe
Joe, I am going to pick a number between 1 and 100.
You must try to guess the number
OK .. I have picked a number.
What is your guess, Joe:  50
That is TOO LOW, Joe
What is your guess, Joe:  80
That is TOO HIGH, Joe
What is your guess, Joe:  60
That is TOO HIGH, Joe
What is your guess, Joe:  55
That is TOO LOW, Joe
What is your guess, Joe:  58
Joe, you got it: SPOT ON. My number was 58
You took 5 guesses .. that is PRETTY GOOD.
==================
Does anyone else want to play (y/n) ?  y
 Enter Your Name:    Fred
Fred, I am going to pick a number between 1 and 100.
You must try to guess the number
OK .. I have picked a number.
What is your guess, Fred:  75
Fred, you got it: SPOT ON. My number was 75
You took 1 guess .. that is UNBELIEVABLY BRILLIANT
==================
Does anyone else want to play (y/n) ?  n
Thanks for playing!
2 players have played this session
==================
```

3.6 Game Choices

Write and test the opening section of a computer game program that can be played at 4 skill levels.

a) First the program inputs the player's name.
b) Then the player selects their skill level as follows: -
   Four skill choices are displayed.
   1: Advanced
   2: Experienced
   3: Average
   4: Novice
   The player is asked to select one of these (by entering a number 1 - 4).
   Any wrong choices (e.g. 5 or 6) cause an error message to be displayed.
c) The player is then asked to confirm that their skill choice is OK.
d) The player inputs a reply of 'y' or 'n' .
e) The skill choice selection is validated by repeating until the player's reply is 'y'.
f) At the end of the program the skill choice and player name are displayed with an appropriate message.

```
Brian's SuperDuper Game
==========
Enter Your Name:    Sally
Sally, there are 4 skill levels in this game:
        1. Advanced
        2. Experienced
        3. Average
        4. Novice
What skill level do you choose?  5
Sorry, Sally …you should choose 1-4:
What skill level do you choose?  7
Sorry, Sally …you should choose 1-4:
What skill level do you choose?  3

Thank you Sally, you have chosen level 3.
Is this what you want (y/n)?  n

Sally, there are 4 skill levels in this game:
        1. Advanced
        2. Experienced
        3. Average
        4. Novice
What skill level do you choose?  2

Thank you Sally, you have chosen level 2.
Is this what you want (y/n)?  y
Good:  Sally and you have chosen level 2 and you can now start the game!
```

EXTRA: Can you now link 3.5 and 3.6 together into one game?