code.sprint MT

Preparatory Workbook

Your starting point towards Codesprint Malta

DIRECTORATE FOR LEARNING &
ASSESSMENT PROGRAMMES

Table of Contents

Welcome to CODESPRINTmt

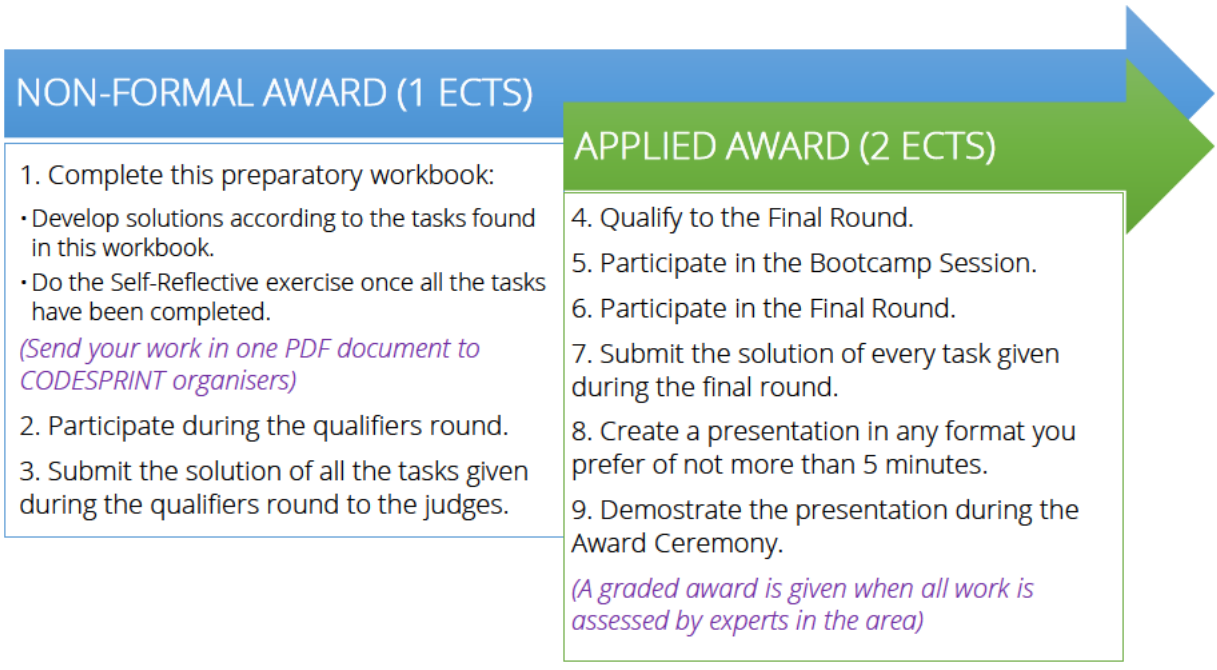
Join CODESPRINTmt, a timed coding challenge that puts your problem-solving / coding skills to the test. Besides the coding fun, you will have the opportunity to meet coding enthusiasts and sharpen your coding competitive skills while having the chance to win amazing prizes.

CODESPRINTmt is divided into two phases: 1) Qualifiers Round, and 2) Finals Round. Half the registered participants (maximum of 20 participants) will pass on to the final round.

During both rounds:

- you will be given a series of tasks ordered by level of difficulty.
- every task is timed according to the level of difficulty.
- you are to carry out the tasks on an individual basis.
- you must submit your solutions to the judges at the end of every task.
- a judging panel will give points to your work according to an established criteria rubric.
- a detailed assessment rubric will be given during the contest. Therefore, you will be aware of what the judges are expecting from your work.

Moreover, CODESPRINTmt is accredited at MQF Level 3, thus offering you the opportunity to top up your list of academic achievements. You can be awarded either a **1 ECTS Non-Formal MQF3 Award** (not graded) or a **2 ECTS Applied MQF3 Award** which is graded as explained in section Grading Distribution.

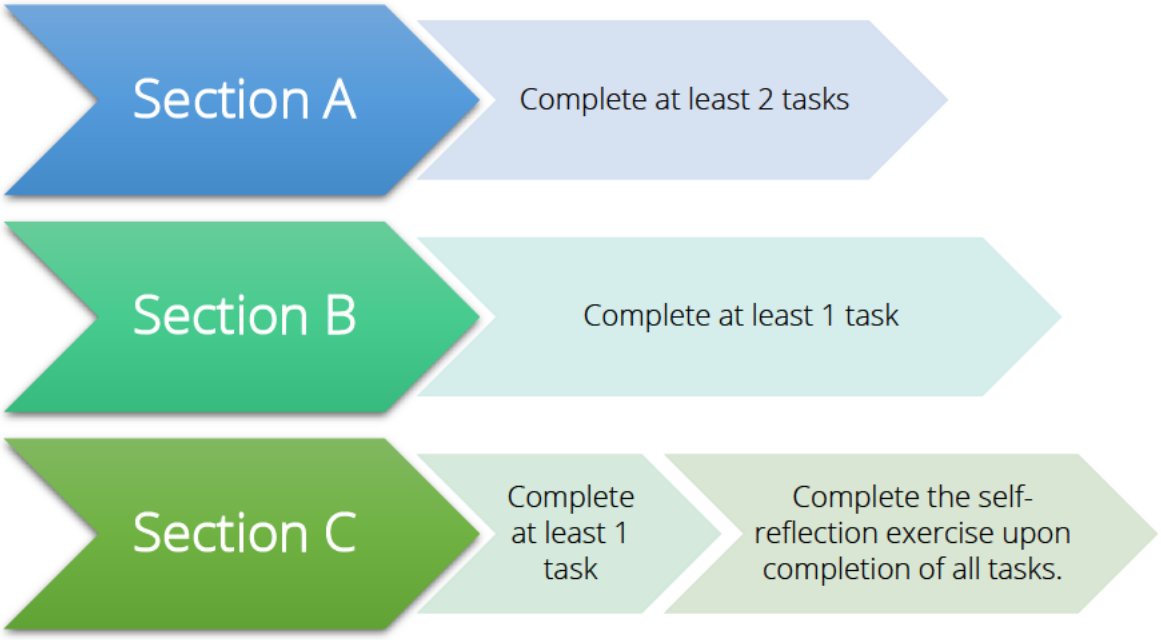


Workbook Guidelines.

This workbook includes:

- Three tasks in Section A
- Two tasks in Section B
- Two tasks in Section C
- Assessment Criteria for every task
- Self-Reflection Exercise
- Guidelines for the Final Presentation *(requirement for the Applied MQF3 Award)*

Participants are expected to complete at least **FOUR** tasks from this workbook, as indicated in the diagram below. Besides the development of the source code, participants are also expected to complete the self-reflection exercise when the tasks are completed.



ALL WORK MUST BE SAVED IN **ONE ZIP FILE** AND SENT BY THE CLOSING DATE, AS INSTRUCTED IN THE RULES AND REGULATIONS BOOKLET

The Zip filename should be your name_surname_school.pdf
For example: alexia_brincat_stignatiuscollege.zip

Grading Distribution

The 2 ECTS Applied MQF 3 Certification is graded as follows:

0 - 39	40 - 59	60 - 89	90 - 100
FAIL	PASS	MERIT	DISTINCTION

The marks from each phase of the event contribute to a global mark. The distribution of marks is as flows:

Section A

Task 1: Automated Teller Machines

Automated teller machines (ATM) require the user to enter a PIN Number to access the bank card. Write a program that simulates this part of the ATM program.

The program should ask the user to enter a four-numbered PIN code. If it matches the PIN code assigned to the Bank Card, it displays *"CORRECT PIN";* otherwise it displays *"INCORRECT PIN"*. The program should warn the user that the ATM will hold the bank card by displaying *"BANK CARD HELD"* upon entering the wrong PIN code for three (3) consecutive times.

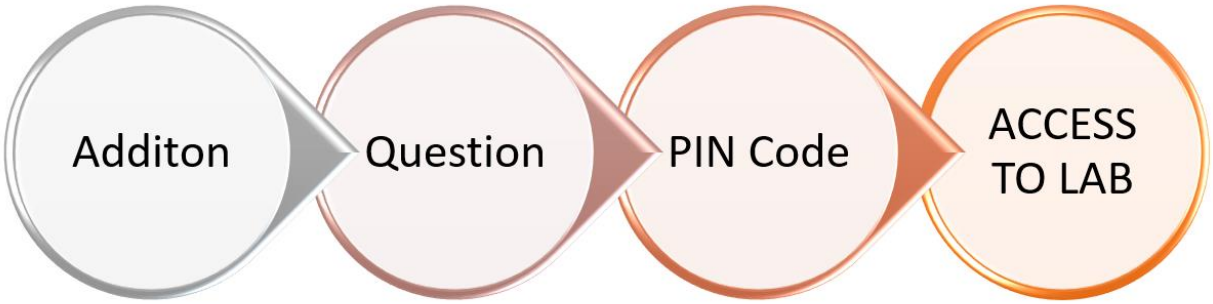
Program Rules:

- PIN assigned to Bank Card is a constant value - 9681.
- If the PIN code entered does not match the proper format (not a 4-numbered pin), the program displays *"INVALID PIN FORMAT"* and does not count the entry as an incorrect PIN.
- It is assumed that only numbers are entered by the user since only numpads are usually available on ATMs

1. Name the class containing the main method RunApp1
2. Submit your program in a folder called Task1_Name; *e.g.* Task1_ John

Task 2: Nuclear Power Plant Access Area

A scientist can access the Nuclear Powerplant lab by going through a three-tiered validation protocol as in the diagram below:



Step 1: Addition

The scientist must answer correctly a randomly generated addition problem with two numbers. The values to be randomised are between 0 and 9; e.g. 3+7 = ? where 3 and 7 are randomly generated.

Step 2: Question

The scientist must answer correctly a multiple-choice-answer question which is randomised from a list of three pre-set questions. The pre-set questions & answers are listed below (pg. 4); the correct answers are enclosed in a blue box.

Pre-set Questions

Question 1:

Which element is used as fuel in a nuclear power stations?

A: Water	B: Gas	**C: Uranium**

Question 2:

Which country uses the most nuclear power?

A: The United States	B: Russia	C: France

Question 3:

Which country opened the first nuclear power plant in 1954 known as 'Atom Mirny'?

A: North Korea	**B: The Soviet Union**	C: Japan

Step 3: PIN Code

The scientist must enter a four-numbered PIN Code which is the constant value 6502.

Log-In Access Flow

From one validation step to another the program will not provide any feedback to the scientist. At the end of the validation process, the scientist will be granted or denied access to the lab by displaying a message accordingly.

Hint:

To ignore case sensitivity for the user's answer (char), the user's input can be changed into its uppercase equivalent. For example, if the user enters character 'b', it can be changed to character 'B'. This can be done using the code:

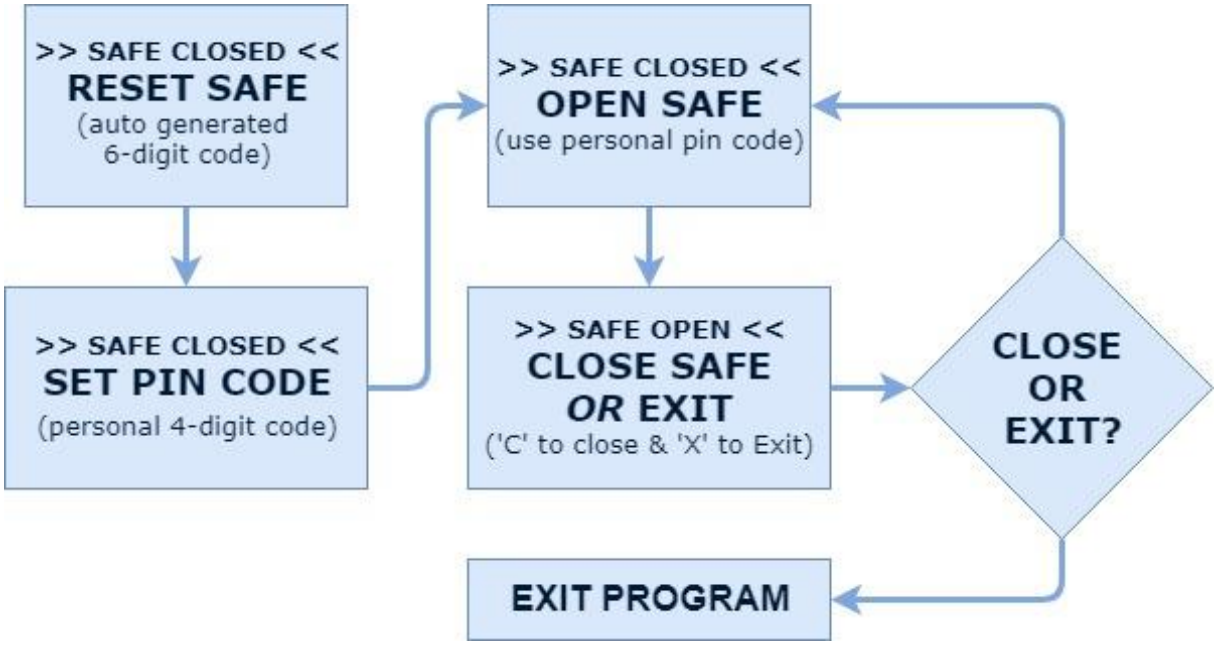
```
userInput = Character.toUpperCase(userInput);
```

1. Name the class containing the main method RunApp2
2. Submit your program in a folder called Task2_Name; *e.g.* Task2_ John

Task 3: Hotel Room Safe Box

Hotel guests should feel safe leaving their belongings in their room. Hotels from all around the world, nowadays offer a safe box in every room. Guests can therefore use the safe box to secure any valuable personal belongings.

The safety box has a keypad so that the user can use a personal pin number to open and close the safe. Write a program that simulates a safe box which works as follows:



Program Rules:

- The guest must reset the safe box by using a 6-digit random pin code which is auto generated and displayed upon initialization of the program. *A proper message is displayed if a wrong pin, or a non-6-digit pin is entered.*
- After resetting the safe box, the guest should enter a personal 4-digit pin. *A proper message is displayed if a non-4-digit pin is entered.*
- The user can open the safe box by entering the personalised pin code. *A proper message is displayed if a non-4-digit pin is entered.*
- The user can close the safe box by entering 'C' or exit program by entering 'X'. *A proper message is displayed if the character 'C' or 'X' is not used to close the safe box or exit program respectively.*
- Proper indications of the status of the safe box should be displayed with every step.

1. Name the class containing the main method RunApp3
2. Submit your program in a folder called Task3_Name; *e.g.* Task3_ John

Section B

Task 4: Lotto Ticket

The lottery is a game in which players pay for a ticket, select a group of numbers and win prizes based on how they match the drawn results.

Write a program that simulates a lottery system according to rules below:

Program Rules:

- The lottery system has a set of numbers available which is from 1 to 45.
- The lottery prize is that of €500,000.
- Five lottery numbers are automatically drawn and are not visible to the user. *Proper validation is required to avoid duplicate numbers.*
- The user is asked to purchase a lottery ticket.
- A lottery ticket is made up of five non-duplicate numbers. *Proper validation is required to avoid non-valid and duplicate numbers.*
- According to the numbers guessed, a prize is won:
  - With three numbers guessed, the user wins 10% of the lottery prize.
  - With four numbers guessed, the user wins 25% of the lottery prize.
  - With five numbers guessed, the user wins lottery prize in full.
- The result-screen should display the numbers drawn, the amount of numbers guessed, and the prize won (if applicable).

1. Name the class containing the main method **RunApp4**
2. Submit your program in a folder called **Task4_Name**; *e.g.* Task4_ John

Task 5: Point-of-Sale System

POS systems have replaced most traditional cash registers due to their ability to connect to the retailer's main database system. Having all data stored and accessible within one system makes daily operations more efficient and more profitable.

Write a program that simulates a POS system. This system has a Log-In Screen and a Main Menu as shown below:

Log In Screen	
Option	**Description**
Cashier Log In	A cashier must be logged-in to proceed to the Main Menu. *(Details of the registered cashiers is shown in **Table 2**)*
Exit	Terminates the program.

Main Menu	
Option	**Description**
Enter new transaction	Allows the cashier to enter the items that the client wants to buy. *(A list of the stock items is shown in **Table 1**)*
Issue Receipt	This option displays the receipt of the last transaction carried out. *(A sample is shown in **Figure 1**)*
Display Stock List	This option displays the list of items that a client can buy. *(A list of the stock items is shown in **Table 1**)*
Cashier Sign Out	Allows a logged-in cashier to sign out and return to the Log-In Screen. *(Details of the current registered cashiers is shown in **Table 2**)*

Stock List	
Items	Price
Printer	€67.99
Monitor	€138.00
Keyboard	€12.50
Graphics Card	€114.99
Soundbar	€249.00
Hard Disk	€66.95
Headset	€17.55
Smartwatch	€135.00
Camcorder	€329.00
Drone	€449.99

Table 1

```
-----------------------------------
 << EASY-SAVE SUPERMARKET >>
 102, Flower Street, Filfla
***********************************
 Sun Jan 20 11:48:45 CET 2019
 Receipt: 66432020
 Cashier: mallia.amy
***********************************
 MONITOR                 Eur 138.00
 HEADSET                 Eur 17.55
 SOUNBAR                 Eur 249.00


 SUBTOTAL:               EUR 404.55
 VAT:                    Eur 72.82
 TOTAL:                  Eur 477.37

***********************************
          FISCAL RECEIPT
            THANK  YOU
------------ CUT-HERE --------------
```

Figure 1: Sample Generated Receipt

Registered Cashiers	
Username	Password
borg.steve	Borg87max
zammit.rita	RitPopSing!
agius.john	ToyotaBeSt
vella.carlos.02	Rock!n!Roll
mallia.amy	$Gaga$Lady$

Table 2

1. Name the class containing the main method **RunApp5**
2. Submit your program in a folder called **Task5_Name**; *e.g.* Task5_ John

Hints:

1. To ignore case sensitivity of the username (String), the user's input can be changed into its uppercase equivalent. For example, if the user enters username 'borg.joe', it can be changed to 'BORG.JOE'. This can be done using the code: `userInput = userInput.toUppercase();`

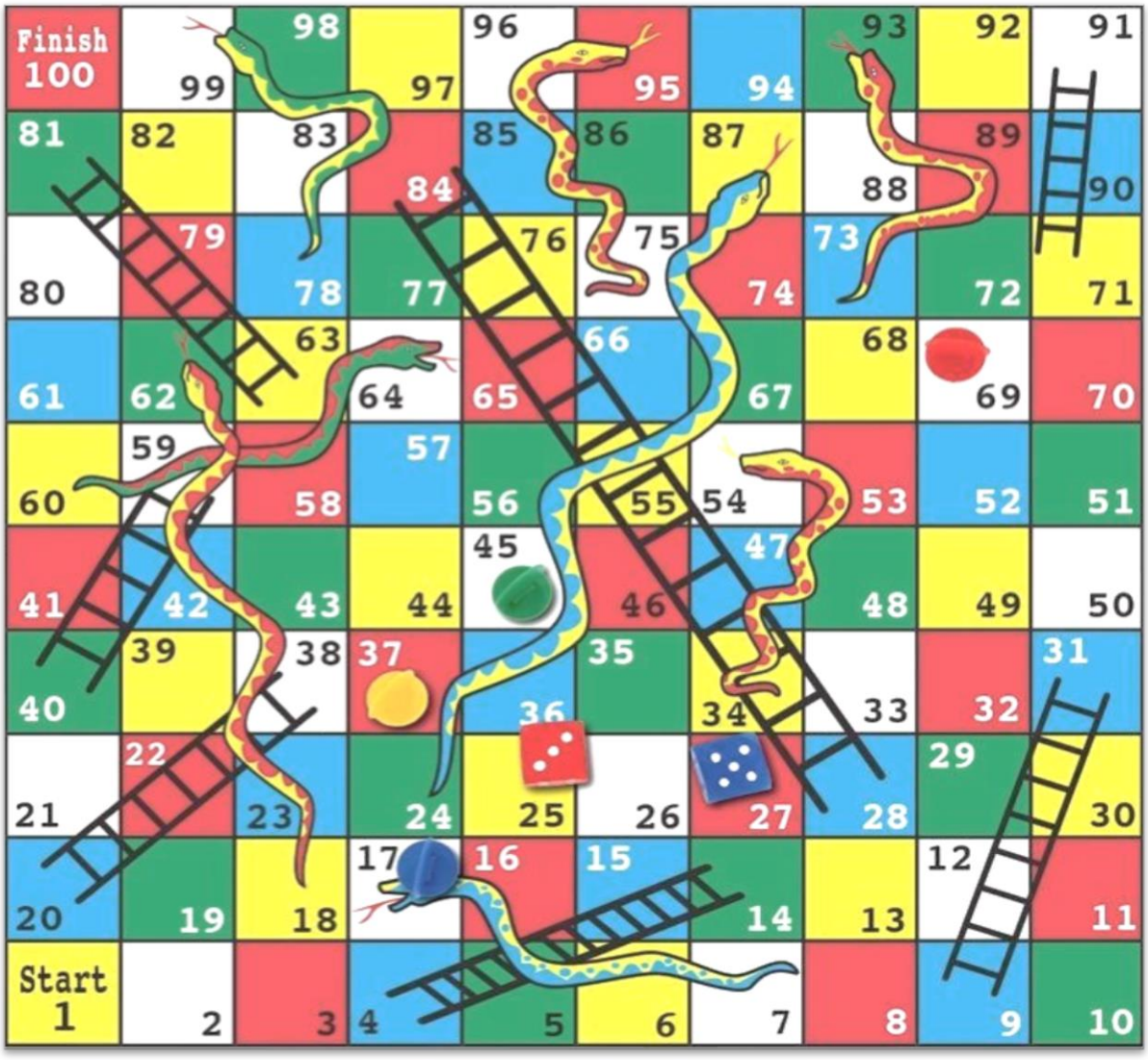
2. When comparing a String variable in a conditional statement the **`.equals()`** method should be used.
   - ⇨ **`if(userInput == "BORG.JOE"){}`** will not work because a String variable cannot be compared using **==**.
   - ⇨ The correct version is: **`if(userInput.equals("BORG.JOE")){}`**

3. To display the date:
   a. the library **`java.util.Date`** needs to be imported
   b. an instance of class Date should be created: **`Date myDate = new Date();`**
   c. display the date: **`System.out.println(myDate.toString());`**

Section C

Task 6: Snakes & Ladders

Snakes and Ladders is an ancient Indian board game regarded today as a worldwide classic. It is played between two or more players on a gameboard having numbered, gridded squares starting from 1 to 100. Many "ladders" and "snakes" are pictured on the board, each connecting two specific board squares as shown in the diagram below. The objective of the game is to navigate, according to die rolls, from the start (Step 1) to the finish (Step 100), helped or hindered by ladders and snakes respectively.



The game is a simple race contest based on sheer luck. The historic version had roots in morality lessons, where a player's progression up the board represented a life journey complicated by virtues (ladders) and vices (snakes).

Program Rules:

- The game must be played in TWO (2) players mode.
- Players' name must be entered before the game starts.
- Player 1 starts playing and player 2 follows until one of the players reaches step 100.
- When one of the players reaches step 100, the program displays the name of the winner.
- Player 1 can roll the dice by pressing 'X'.
- Player 2 can roll the dice by pressing 'Z'.
- The dice provides a random number from 1 to 6.
- If the dice gets the maximum value (6), the player HAS AN EXTRA CHANCE to play before the other player continues.
- If the player lands on a step that contains a ladder or a snake, THE PLAYER'S POSITION WILL CHANGE according to the diagram on page 13, or as simplified in the table below.
- During the game PROPER INDICATIONS of the players' status should be displayed.

Player's Position	Element	New Position
4	Ladder	14
9	Ladder	31
17	Snake	7
20	Ladder	38
28	Ladder	84
40	Ladder	59
54	Snake	34
62	Snake	18

Player's Position	Element	New Position
63	Ladder	81
64	Snake	60
71	Ladder	91
87	Snake	24
93	Snake	73
95	Snake	75
99	Snake	78

1. Name the class containing the main method **RunApp6**
2. Submit your program in a folder called **Task6_Name**; *e.g.* Task6_ John

Task 7: Mastermind

Mastermind is a code-breaking game invented in 1970 by Mordecai Meirowitz, an Israeli postmaster and telecommunications expert.

In this version of MASTERMIND, the computer has the role of ***codemaker*** and the player that of ***codebreaker***. The codemaker randomises a pattern of four coloured pins from six available colours. This colour pattern is hidden from the codebreaker. The colour pattern can contain colour duplicates; for instance, the pattern could be four of the same colour! The available colours are:

BLUE – GREEN – PURPLE – RED - YELLOW - WHITE

The codebreaker tries to guess the pattern, in both order and colour, within ten turns. With every guess the codemaker provides feedback by displaying the number of correct coloured pins guessed and the number of correct pin positions, as shown in the examples below:

	Example 1			
Code	RED	BLUE	PURPLE	BLUE
User Guess	RED	WHITE	BLUE	BLUE
Feedback	Pins Guessed: 3 Correct Pin Positions: 2			

	Example 2			
Code	WHITE	BLUE	YELLOW	RED
User Guess	RED	RED	GREEN	YELLOW
Feedback	Pins Guessed: 2 Correct Pin Positions: 0			

Program Rules:

- The codemaker randomises a pattern of four coloured pins from the six available colours. *The generated code can contain pins of the same colour.*
- The codebreaker has ten chances to guess the pattern correctly.
- With every guess, the codemaker provides proper feedback; i.e. number of correct pins guessed and number of correct positions.
- During the game proper indication of the player's number of chances should be displayed. *See sample interface on the right.*
- The codebreaker wins if the pattern is guessed without losing all ten chances.
- The codemaker wins if the codebreaker loses all ten chances without guessing the code successfully. In this case the generated colour code will be displayed.

```
<<<<<<<<<<<<<< Guess the 4 colours >

[W]hite, [R]ed, [G]reen, [B]lue, [Y]e

----------------- YOU HAVE 8 LIVES --

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

Colour 1: r
Colour 2: r
Colour 3: g
Colour 4: y

CODE GUESSED: R R G Y >>>
<<<< RESULT >>>>
Pins Guessed:  2
Correct Pins Positions:  1

********************
*    TRY AGAIN     *
********************
```

Screenshot: Sample Interface

1. Name the class containing the main method **RunApp7**
2. Submit your program in a folder called **Task7_Name**; *e.g.* Task7_ John

Assessment Criteria

Automated Teller Machine				
Program Functionality	User Friendly Interface	Code Efficiency	Proper use of In-line Text *(Comments)*	Use of proper Conventions
Name of Folder & Class/es	Constant Declaration & Initialisation	User Input	Suitable Prompts/Messages displayed	Validation of PIN Code *(4 Digits only)*
Extra Features *(not listed in the task)*	**Maximum Score:** 20 + 2 for every extra feature			
0 – Not Satisfactorily | 1- Partly Satisfactorily | 2- Entirely Satisfactorily				

Nuclear Power Plant Access Area					
Program Functionality	User Friendly Interface	Code Efficiency	Proper use of In-line Text *(Comments)*	Use of proper Conventions	Name of Folder & Class/es
Constant Declaration & Initialisation	User Input	Suitable Prompts/Messages displayed	Proper Randomisation of Addition Problem	Proper Randomisation of Questions	Proper Validation of User's Answer *(A, B or C only)*
Proper Validation of PIN Code *(4 Digits Only)*	Extra Features *(not listed in the task)*	**Maximum Score:** 26 + 2 for every extra feature			
0 – Not Satisfactorily | 1- Partly Satisfactorily | 2- Entirely Satisfactorily					

Hotel Room Safe Box						
Program Functionality	User-Friendly Interface	Code Efficiency	Proper use of In-line Text (Comments)	Use of Proper Conventions (Camel Case, meaningful variable names etc.)	Name of Folder & Class/es	User Input
Suitable Prompts / Messages displayed	Proper Randomisation (6-digit pin code)	User's input Validation (pin codes, close safe box & exit program)	Other Features (not listed in the task)	**Maximum Score:** 20 + 2 for every extra feature		
0 – Not Satisfactorily | 1- Partly Satisfactorily | 2- Entirely Satisfactorily						

Lotto System				
Program Functionality	User Friendly Interface	Code Efficiency	Proper use of In-line Text *(Comments)*	Use of Proper Conventions *(Camel Case, meaningful variable names etc.)*
Name of Folder & Class/es	User Input	Suitable Prompts / Messages displayed	Numbers Drawn Automatically *(5 numbers)*	Validation *(non-duplicates & valid numbers)*
Arithmetic Calculations *(total numbers guessed & prize won)*	Proper Use of Data Structure *(such as Arrays)*	Other Features *(not listed in the task)*	Maximum Score: 24 + 2 for every extra feature	
0 – Not Satisfactorily | 1- Partly Satisfactorily | 2- Entirely Satisfactorily				

Point of Sale System				
Program Functionality	User Friendly Interface	Code Efficiency	Proper use of In-line Text (Comments)	Use of Proper Conventions (Camel Case, meaningful variable names etc.)
Name of Folder & Class/es	User Input	Suitable Prompts / Messages displayed	Options Validation (Login Screen, Main Menu & New Transaction)	Functionality Validation (Issuing of Receipt)
Ignoring Case Sensitivity (when searching for username & item)	Proper Use of Data Structure (such as Arrays)	Searching of Records (Stock & Cashiers)	Arithmetic Calculations (Subtotal, VAT & Total)	Generating Receipt Number (8-digit Number)
Display Receipt (simulating a real receipt as much as possible)	Display list of items in stock (including formatting)	Other Features (not listed in the task)	Maximum Score: 34 + 2 for every extra feature	
0 – Not Satisfactorily | 1- Partly Satisfactorily | 2- Entirely Satisfactorily				

Snakes & Ladders				
Program Functionality	User Friendly Interface	Code Efficiency	Proper use of In-line Text *(Comments)*	Use of Proper Conventions
Name of Folder & Class/es	User Input	Suitable Prompts/Messages displayed	Proper Randomisation of the dice	Proper Validation of the User's input to roll the dice *(X or Z only)*
Ignoring Case Sensitivity *(User's Input to Roll the Dice)*	Other Features *(not listed in the task)*	Maximum Score: 22 + 2 for every extra feature		
0 – Not Satisfactorily | 1- Partly Satisfactorily | 2- Entirely Satisfactorily				

Mastermind				
Program Functionality	User Friendly Interface	Code Efficiency	Use of In-line Text *(Comments)*	Use of Proper Conventions *(Camel Case, Variable names etc.)*
Name of Folder & Class/es	User Input	Suitable Messages displayed	Proper Randomisation *(Colour Pattern)*	Proper Validation of the User's input *(Colours chosen)*
Ignoring Case Sensitivity *(when guessing colours)*	Proper Use of Data Structure *(such as ARRAYS)*	Other Features *(not listed in the task)*	Maximum Score: 24 + 2 for every extra feature	
0 – Not Satisfactorily | 1- Partly Satisfactorily | 2- Entirely Satisfactorily				

Self-Reflection Exercise

The following ten questions are intended to help you reflect on the work that you did and the effort that you invested in preparation for CODESPRINTmt.

Answer all questions:

1. To what extent did you manage to complete the tasks? Rate 1-5.
   [1 being all tasks were not successfully finished and 5 being all tasks were successfully finished]
2. What did you learn from completing these tasks?
3. What did you like most about these tasks? Why?
4. What difficulties did you encounter in tackling these tasks? How did you solve them?
5. Which tasks / part of tasks did you find most challenging?
6. What level of support from your mentor did you seek in completing the tasks to the level you did?
7. What other resources have you used in completing this workbook?
8. Mention some testing procedures that you carried out for each task.
9. Suggest further improvements to your work.
10. What did you learn from completing this workbook?

Each question is awarded 10 marks, of which:

- 7 marks are given for the content, and
- 3 marks for using a reflective rather than a descriptive approach.

Final Presentation Guidelines

Prepare a presentation to briefly outline your CODESPRINT journey including:

- An introduction of:
  - yourself,
  - your coding backgrounds,
  - your past CODESPRINT experiences (if any).
- Explain your choice of workbook tasks.
- The problem-solving strategies that you employed in the different tasks.
  - If any, identify situations where you have found more than one solution to tackle part/full task and on what grounds did you choose one strategy over another.
- The difficulties you encountered, and which were:
  - solved and how did you overcome your difficulties,
  - not solved.
- The lessons you have learned from this experience.
- Your suggestions to future candidates.
- Your suggestions to CODESPRINT organisers.

Your presentation will be assessed as follows:

40%	30%	15%	10%	5%
Showing a reflective approach, not just demonstrating a descriptive journal of your experience.	Using at least three different sources of support, such as mentor s support and online sources.	Answering all above questions.	Completing your presentation in maximum of five minutes.	Showing a sense of originality in your presentation.

Your presentation:

- can take any format that you prefer such as a PowerPoint, a Word document, a video, a website etc.,
- should not exceed five minutes,
- should be presented during the CODESPRINT Award Ceremony.

NOTES

Official Event Contacts:

Mr. Josmar Borg – Education Officer (Computing)

21422337 / 21431408

josmar.borg@ilearn.edu.mt

A project created by the Computing Department,
within the Directorate for Learning and Assessment Programmes (DLAP)

Directorate for Learning and Assessment Programmes, MEDE