UTM UNIVERSITI TEKNOLOGI MALAYSIA

SCHOOL OF COMPUTING
Faculty of Engineering

UNIVERSITI TEKNOLOGI MALAYSIA

MID-TERM TEST SEMESTER II, 2021/2022

SKILL-BASED TEST (PRACTICAL)

SUBJECT CODE	**:**	**SECJ1023 / SCSJ1023**
SUBJECT NAME	**:**	**PROGRAMMING TECHNIQUE II**
TIME / DURATION	**:**	**10:00 AM – 11:50 AM (1 HOUR 50 MINUTES)**
DATE/DAY	**:**	**27 MAY 2022 (FRIDAY)**

GENERAL INSTRUCTIONS

- This is an **OPEN-BOOK** test. You can refer to any resources online or offline.
- However, any kind of discussions whether verbal, chats, online, etc. is strictly prohibited.
- This is a practical test. You will attempt the test by doing coding on a computer.
- You will need to video record your attempt answering this test.
- Use e-Learning only for downloading the question and submitting the answers.
- Do coding offline using any C++ IDE such as VS Code, DevCPP, etc.
- **All COMMENTS** you write in the program **WILL NOT BE GRADED**.
- Programs that **CANNOT COMPILE** will get a **PENALTY**.
- Any question related to the test question will not be entertained at all. Read the question carefully.
- The given duration is entirely for coding.
- Be warned that your program will undergo a screening process for **PLAGIARISM DETECTION**.
- Programs that are found cheating will be automatically **rejected**, thus will get a **ZERO** mark.

VIDEO RECORDING INSTRUCTIONS

- Record your coding session from the beginning to the end. Submission without the video will be **DECLINED**.
- You can use Webex software to video record your session.
- Note that, a free account Webex only opens for 50 minutes per session. Thus, should you need more time, you will need to open another session once the current one ends.

- Free account Webex also allows only local recording, i.e. the video will be stored on your computer. Thus, later you will need to upload the videos to the cloud (e.g., to Google Drive) manually.
- Alternatively, you can video record your session using software like Open Broadcaster Software (OBS) or any other tools that works for you.
- The video is meant only for recording your progress in attempting this paper, rather than a presentation.
- You must turn the camera and microphone on for the entire session.
- In the video you should show your VS Code and the output (console terminal).
- If you upload multiple videos on Google Drive, put them in a single folder, and submit only the folder's link. Set the video file (or folder) permissions so that "**Anyone can view**".
- You must not edit the video. Submit it as is.

SUBMISSION INSTRUCTIONS

- This test requires two types of submissions:
- Source code – to be submitted twice as follows:
  - Interim submission at **10:45AM**, will be open for 10 minutes.
  - Final submission at **11:40AM**, will be open for 10 minutes.
- Recorded video – you will be given extra time until, **29 May 2022, 12:00 noon** to submit the video. Upload the video to your own Google Drive account and submit only the video link.
- All submissions must be done on UTM eLearning.

Question [40 Marks]

A straight line is expressed by the equation, $y = mx + c$, where ***m*** is the slope of the line and ***c*** is the *y*-intercept, i.e. the *y* coordinate of the location where the line crosses the y-axis (see Figure 1).

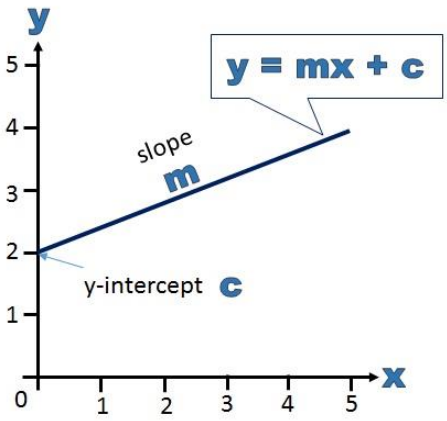

Figure 1

Define a class named `Line` to model a line. Use inline style to define the class. Use the provided template file, **sbt.cpp** to write your code. The requirements for the program are as follows:

1. Define a constructor that accepts two parameters to set the attributes *m* and *c*, respectively. Declare this constructor such that it can also serves as a default constructor (or default arguments constructor). In this case, it will model the line, ***y=x***.

   (3 marks)

2. Define a function named **`read()`** that reads input from the keyboard to set the attributes *m* and *c*.

   (3 marks)

3. Define an overloaded operator that will be used for operations such as

   `line1 - line2`

   where ***line1*** and ***line2*** are both `Line` objects. This operator returns another `Line` object with the slope equal to the slope of ***line1*** subtracted by the slope of ***line2***; and the y-intercept equal to the y-intercept of ***line1*** subtracted by the y-intercept of ***line2***. For example, if ***line1*** represents the line equation $y=5x+2$ (i.e, $m = 5$ and $c = 2$), and ***line2*** represents the line equation $y=x+5$ (i.e, $m = 1$ and $c = 5$) then

line1 - line2

will result in a new line, ***y=4x-3*** (i.e, *m* = *4* and *c* = *-3*).

(3 marks)

4. Define another overloaded operator that will be used for operations like:

line1 != line2

where ***line1*** and ***line2*** are both `Line` objects. This operator returns `true` if the slope of ***line1*** multiplied by the slope of ***line2*** is not equal to -1; otherwise, it returns `false`.

(4 marks)

5. Define a method named **`toString()`** that returns a line equation of type `string`. The following table shows several example values of the attributes *m* and *c*, and the string equation that this method should return. **Note:** Assume ***m ≠ 0***. That means, you do not have to consider cases like ***y=-5***, and ***y=2***.

Line attributes	**Returned Equation**
m = 5, *c* = 2	y=5x+2
m = 1, *c* = -8	y=x-8 *(not y=1x+-8)*
m = -1, *c* = 0	y=-x *(not y=-1x+0)*

(10 marks)

6. Define a regular function named **`printLines()`** and make it as a friend to the class. This function accepts a list of `Line` objects as a parameter and prints out the attributes *m* and *c*, and the string equation of each line in the list.

(5 marks)

7. In the main function, declare an array of `Line` objects of three elements. Initialize the first line to model the equation **y=x+5** and let the remaining lines initialized with the default constructor. Then, print out all the lines.

(3 marks)

8. Write code that makes changes on the second and third elements of the array:
    a. Change the second line using user input. The user needs to enter new values for *m* and *c* of that line.
    b. Change the third line by substracting the first line with the second line.
    c. Then, print out all the lines again.

(6 marks)

9. Write code to determine if the first and last lines are perpendicular to each other. If the product of the two slopes is equal to -1, the two lines are perpendicular. Use the overloaded operator (!=) function to accomplish this task.

(3 marks)

Figure 2 shows two example runs of the program. The **bold text** indicate user inputs.

```
ORIGINAL LINES
Line 1 slope(m) = 1, y-intercept(c) = 5, equation: y=x+5
Line 2 slope(m) = 1, y-intercept(c) = 0, equation: y=x
Line 3 slope(m) = 1, y-intercept(c) = 0, equation: y=x

Set the second line from user input
Enter the slope (m) and y-intercept of a line (c) => 2 8

UPDATED LINES
Line 1 slope(m) = 1, y-intercept(c) = 5, equation: y=x+5
Line 2 slope(m) = 2, y-intercept(c) = 8, equation: y=2x+8
Line 3 slope(m) = -1, y-intercept(c) = -3, equation: y=-x-3

The first line is perpendicular to the last line
```

(a) Run 1

```
ORIGINAL LINES
Line 1 slope(m) = 1, y-intercept(c) = 5, equation: y=x+5
Line 2 slope(m) = 1, y-intercept(c) = 0, equation: y=x
Line 3 slope(m) = 1, y-intercept(c) = 0, equation: y=x

Set the second line from user input
Enter the slope (m) and y-intercept of a line (c) => 3 5

UPDATED LINES
Line 1 slope(m) = 1, y-intercept(c) = 5, equation: y=x+5
Line 2 slope(m) = 3, y-intercept(c) = 5, equation: y=3x+5
Line 3 slope(m) = -2, y-intercept(c) = 0, equation: y=-2x

The first line is not perpendicular to the last line
```

(b) Run 2

Figure 2