The University of Nottingham Ningbo China

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

A LEVEL 1 MODULE, 2022-2023

EEEE1044 Introduction to Software Engineering and Programming

Time allowed: **TWO Hours**

Candidates may log in to computers and test CodeBlocks and sign their desk card but must NOT write anything else until the start of the examination period is announced.

Answer ALL Questions

Only a calculator from approved list A (or one functionally equivalent) may

be used in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a dictionary to translate between that language and English provided that neither language is the subject of this examination.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

Do read the Exam Procedure available on desktop before starting the exam.

ADDITIONAL MATERIAL: Exam Procedure

INFORMATION FOR INVIGILATOR:
None.

Q1. Develop a program which meets all the following requirements.

a) Generate **ONE** random integer number in the range from 0 to 24 (inclusive).

b) Display the corresponding message for the random number following the information given in Table Q5.

Table Q1

From 0 to 7 and from 22 to 24	Bed Time
8, 9, 10, 11, 12	Morning!
13, 14	Get Fuel!
15, 16, 17, 18	Keep fighting
19, 20, 21	Fun Time

Marking Scheme 6

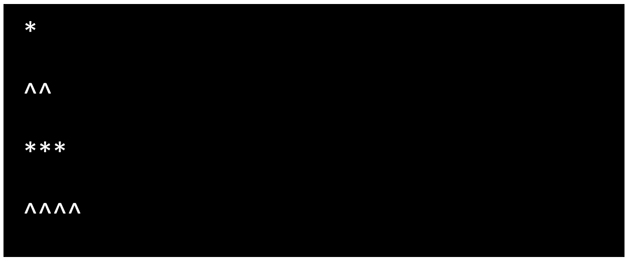
Item	Mark
Comment	2
Generate rand number	2
Display correct message (full mark for switch case, if-else, half mark for multiple if)	2

Q2. Develop a program to print a triangle pattern, which meets all the following requirements.

a) Prompt the user to enter **a positive even** integer, e.g., 2, 4, 10. Check whether the entered input integer satisfies the requirements. If not, prompt the user to input again until a valid input is made.

b) Use the input integer as the **row number** of a **triangular pattern, odd** lines printed with the asterisk symbol (i.e., ‘*****’) and with **even** rows printed with the circumflex symbol (i.e., ‘^’). Take the integer **6** as an example where the pattern that needs to be printed is shown in Figure Q2.

c) Comment your codes properly and use the appropriate indentation.



```
*

^^

***

^^^^
```

Figure Q2

[**20 marks**]

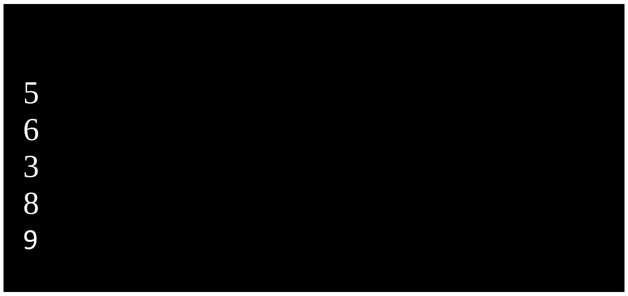
Marking Scheme 20

Item	Mark
Comment	2
Indentation	2
Loop to check positive & even (full mark for while, do-while, 1 mark for if)	6
Display correct pattern (full mark for nested loop, while, or for)	10

Q3. Develop a program which meets all the following requirements.

a) Prompt the user to enter **one 5-digit integer**, e.g. 56389, 12345. If the input does not meet the requirement, display a proper **error message** and prompt the user to enter again, **until a valid input** is entered.

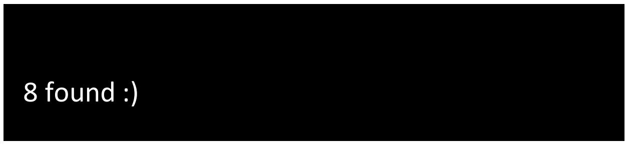
b) Pass the integer to **one external function**. In the external function, display the **5 digits separately with one digit per row**. Taken the input of 56389 as an example, the display of the 5 digits is shown in Figure Q3b.



```
5
6
3
8
9
```

Figure Q3b

c) If there is **one or more 8** in the 5 digits, display the message of ***8 found :)*** as shown in Figure Q3c. Complete this task in the same external function as in step b). *Note: The message* ***should only be displayed once*** *no matter how many 8 are included in the 5-digit integer*.



```
8 found :)
```

Figure Q3c

d) Comment your program properly and use proper indentation.

Marking Scheme 24

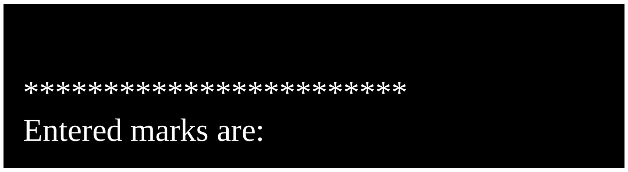
Item	Mark
Comment	2
Indentation	2
Check 5-digit input (as long as it can check if the integer has 5 digits, full mark)	5
External function (full mark for one external function return data type void)	3
Split 5 digits (as long as can split for random 5-digit integer, full mark)	7
Find 8 and display once	5

Q4. Develop a program to collect marks of a group of students, which meets all the following requirements.

a) Declare **an integer array of size 10** for the purpose of mark storage.

b) Use **loop** to prompt the user to input 10 marks. For each mark entered by the user, the program should **check** if the mark is within the range of **0 ~ 100 (both inclusive)**. If the mark entered is out of this range, display a proper **error message** and the invalid mark **should not** be saved into the integer array.

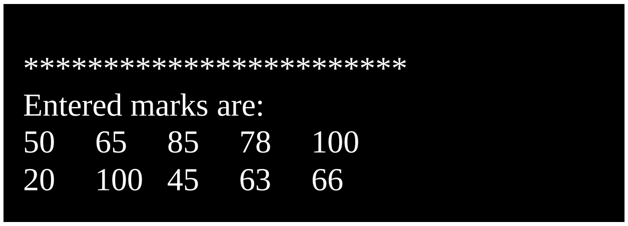
c) After 10 valid marks entered, the program should display the **message** shown in Figure Q2c



```
************************
Entered marks are:
```

Figure Q2c

d) After completing step c), display the 10 valid marks with **a field width of 5, left justified**, and **5 marks per row**. An example display is shown in Figure Q2d. *Note: the display marks of your program is not required to be the same as the integers shown in Figure Q2d*.



```
************************
Entered marks are:
50   65   85   78   100
20   100  45   63   66
```

Figure Q2d

e) Then display the **average mark** of the 10 marks, with **a precision of 2 decimal places**.

f) Comment your program properly and use proper indentation.

Marking Scheme 25

Item	Mark
Comment	2
Indentation	2
Create integer array of size 10	2
Check input range, error message, ignore invalid inputs	5
Use loop to read in 10 marks	5
Display message c)	2
Display 10 marks following requirement of d)	5
Calculate and display average mark	2

Q5. Develop a program to calculate the total credits that one student takes. The text file available on your computer desktop, named "***Credits.txt***" includes the credits of the compulsory modules. Develop your program to meets all the following requirements.

a) One student need to choose **5 optional modules**. Prompt the student to input the credit of each optional module he/she takes in **loop**. The credit should be within the range of **5 ~ 30 (both inclusive)** and should be **divisible by 5**. Check if the input credit is valid. Save the valid input credits into ***Credits.txt*** file **without erasing** the compulsory module credits. *Note: Invalid credits should not be saved into Credits.txt file*.

b) Read all the credits saved in *Credits.txt*, calculate the overall credits (sum) and display **all the credits** and **the overall credits** to the screen. *Note: Make the display to be user friendly*.

c) Comment your program properly and use proper indentation.

Marking Scheme 25

Item	Mark
Comment	2
Indentation	2
Use loop to read in	3
Check range, divisible by 5, don't save invalid input	5
Append credits to file	5
Read from file, calculate and display	8