SEHH2042/SEHS2042 Computer Programming

Individual Assignment 1
Submission deadline: 15 October 2025 (Wednesday), 23:59

Expected Learning Outcomes

- Develop computer programs in one or more high level language programming environment.
- Design and develop structured and documented computer programs.
- Integrate the computer programming techniques to solve practical problems

Introduction

This is an **individual assignment**. There are 3 parts:

Criteria	% of marks
(A) Screencast (Program analysis) *[Q1]*	30
(B) Coding (ShowInfo)	10
(C) Coding (Problem solving) *[Q2 and Q3]*	60

You are given a C++ program template file called *A1Template.cpp*. You are required to **insert C++ codes into the template file** according to the given instructions. Your submission includes (1) a screencast video (in mp4 format), (2) the transcript file of the video, and (3) a complete C++ program which can be compiled and executed successfully. All the requirements stated in this specification need to be satisfied.

Instruction

- For each question, you only need to do **either odd or even version** based on the stated rule about your student ID number (0 is considered as even). You do not need to write any program codes to check the student ID number to determine the version you should do.
- To answer the questions, you need to insert codes into the functions of *the template file*. E.g., to answer question 1, place your code in the scope of *Q1( )*. When the program is executed, enter the question number to run the code of a particular question.
- In Q2 and Q3, the program should first accept input(s) from user **WITHOUT** printing any prompt messages (Hint: *cin* statements will be used, and there is **NO** output messages like "Input n: " before the *cin* statements).
- In Q2 and Q3, you may assume that user always provides valid inputs (correct type and correct range) as required by the question. **NO error input checking** is needed unless otherwise required.
- Follow **EXACTLY** the requirements as stated in the questions. Check the correctness of your code by executing it. In Q2 and Q3, compare its output with the sample display. Test your program carefully by using input test cases as stated in the sample display, as well as other necessary test cases you can think of. Correct output with same output format as the sample display is expected.

- For coding questions, marks are awarded based on the number of test cases that your program can successfully pass without mistakes.
- You may include more header files and write user-defined functions to solve Q2 and Q3. E.g., you may write a user-defined function for solving question 2, and call it in the given function *Q2( )*. ***(refer to "Using template file" on the last page)***
- Apart from inserting codes as mentioned above, you are **NOT** allowed to modify any given codes in the template file.
- **IMPORTANT**: Make sure that your submitted program file **CAN be opened** and has **NO syntax error**. It should be the source code, using .cpp as the file extension. Your submission should **NOT** be other file types such as .zip or .sln.

ShowInfo (10%)

Insert your codes inside the *showInfo( )* function block so that your personal particulars are displayed in the following format when the program executes:

Sample display:

```
Name      : XXX YYY ZZZ
Student ID: 24xxxxxxA
Class     : A01A
```

<u>Question 1 (30%)</u>

Given the program segments below:

Even Version (even 4th digit in student ID)

```
cout << "Q1 Program Output - Even Version\n";
int X = 5;

for (int k = 1; k <= 3; k++) {
     cout << X << ":";
     do {
          cout << "A-";
          X += 2;
     } while (X < 10);
     cout << endl;
}
```

Odd Version (odd 4th digit in student ID)

```
cout << "Q1 Program Output - Odd Version\n";
int Y = 3;

for (int k = 1; k <= 3; k++) {
     cout << Y << ":";
     do {
          cout << "A-";
          Y += 3;
     } while (Y < 10);
     cout << endl;
}
```

You are required to demonstrate your program analysis skills using a screencast video. Based on your student ID number, correctly choose and insert the above program segment **<u>without any modification</u>** into *Q1( )* function block of the program template. Use Microsoft Visual Studio to compile and execute the program successfully. The screencast video you prepare should include your actions to compile and execute the C++ program, and your own explanations about why the program outputs are produced upon program execution.

<u>Detailed requirements of the screencast video</u>

- Duration of video should be between 2 to 4 minutes long. Use **MS Teams** to record. The video should be in mp4 format.
- The video should be recorded to <u>show the computer screen</u> and <u>enable transcription</u>. Show the whole screen, not just part of the screen. While showing the source codes or program outputs, make sure the text is clearly shown, large enough and visible.
- At the beginning, use around 10 seconds to show your personal profile page of the Blackboard platform for identity verification.

- After showing your identity, the video should show the actions you took to <u>compile</u> and <u>execute</u> your program using <u>Microsoft Visual Studio</u>. You also need to show the inserted source codes of the given program segment. However, you do not need to show the actions of typing/ pasting of program segment into the program template file.
- After the program is executed, the video should show clearly the program output window and the text output of the program.
- You should explain the program by describing the program execution flow, how the variable values changed during the loop iteration, giving explanations about how loop(s) repeat and/or stop, and stating the program outputs when *cout* statements are executed.
- Your explanation should be about the given program segment only (not including the template file's main function).
- In your video, use around 10 seconds to show the lecture notes (at most 2 pages) that teach you the programming concepts about relevant loop structures and working principles. The time to show them in the video recording is up to you to decide.
- The video should include your <u>voice</u> recording (in English). Your voice should be clear and loud enough.
- Use suitable cursor movement or text highlighting by mouse action during your presentation.
- You do <u>NOT</u> need to edit the transcript file. The original transcript file produced by MS Teams should be submitted. You can download the transcript file when you play back the video ("view recap").

<u>Hints:</u> To explain the program segment about execution of nested loop, you may find the following dialogs, terms or phrases useful can be used in your presentation:

- *"At the beginning, the value of variable … is …"*
- *"In the outer for-loop, the initial value of k is 1. …*"
- *"In the inner do-while loop, because …"*
- *"The execution enters the loop body because the loop condition is true. …"*
- *"Therefore, … is displayed."*
- *"Variable X is increased by …" ("Variable Y is increased by …")*
- *"Program execution goes to next round."*
- *"The program prints …, then, …"*
- *"The condition check of the loop gives false, which means …."*

<u>Optional task: Early submission of video for initial checks</u>

- It is an optional task. It does not contribute to any marks in the assignment.
- The early submission should be a mp4 video. It will be checked against the following basic requirements <u>ONLY</u>: "show screen", "enable transcription", "include voice", "show identity", "clear and large enough text". No other requirements (such as code explanations) will be checked for you.
- Deadline of early submission: <u>**5 October 2025**</u> (10 days before actual submission deadline). Late submission will not be accepted.
- You will receive the initial checking result <u>**by 10 October 2025**</u> (5 days before actual submission deadline).

Question 2 (30%)

Write your codes inside *Q2( )* function block that calculates the total charge of a taxi ride under a service type (Type A or Type B) according to the table below.

The program should accept the following user inputs **in order**. Note that **NO** prompt messages are needed.

- Traveling distance (in kilometers) (a *double* value)
- Total waiting time (in minutes) (an *integer* value)
- Dynamic pricing mode (1: discount / 2: normal / 3: peak) (an *integer* value)
- Surcharge (in dollars) (an *integer* value)

After receiving the above inputs, the program shows the total charge calculated in **two decimal places**.

The total charge is calculated by adding the **basic charge** with the **surcharge**. The basic charge is determined by the table below, which includes the charges incurred from traveling and waiting.

	Even 5th digit in student ID	*Odd 5th digit in student ID*
Items	**Charge for Type A**	**Charge for Type B**
First 2 km traveling distance (same charge for incomplete 2km)	\$34	\$39
Every subsequent complete 200-metre	\$2.4 (Until the basic charge amount reaches \$130)	\$2.6 (Until the basic charge amount reaches \$130)
	\$1.9 (After the basic charge amount has reached \$130)	\$2.1 (After the basic charge amount has reached \$130)
Every complete 2-minute waiting time	\$2.4 (Until the basic charge amount reaches \$130)	\$2.6 (Until the basic charge amount reaches \$130)
	\$1.9 (After the basic charge amount has reached \$130)	\$2.1 (After the basic charge amount has reached \$130)

Charge adjustment due to dynamic pricing mode (in percentage, see table below) should be applied to basic charge based on different dynamic pricing mode. Note that such adjustment does not affect the surcharge amount in the calculation.

Dynamic pricing mode	**Charge adjustment for Type A and Type B**
1: discount	10% discount on basic charge
2: normal	No adjustment
3: peak	Additional 10% of basic charge

Below shows the sample output display after the program is successfully executed and receives the four user inputs in correct order.

Even Version (even 5th digit in student ID)	*Odd Version (odd 5th digit in student ID)*
Sample display ***3*** ***10*** ***1*** ***5*** `57.20`	**Sample display** ***3*** ***10*** ***1*** ***5*** `63.50`
Sample display ***6.5*** ***8*** ***2*** ***50*** `146.40`	**Sample display** ***6.5*** ***8*** ***2*** ***50*** `156.60`
Sample display ***8.9*** ***17*** ***3*** ***2*** `149.18`	**Sample display** ***8.9*** ***17*** ***3*** ***2*** `161.72`
Sample display ***25.05*** ***0*** ***2*** ***0*** `272.50`	**Sample display** ***25.05*** ***0*** ***2*** ***0*** `298.50`

IMPORTANT NOTE: Compare your program output with the sample display. There should be **NO** prompt messages and **NO** additional blank lines in the output. **DO NOT add extra text** in the output which will be considered as incorrect in marking.

<u>Question 3 (30%)</u>

Write your codes inside *Q3( )* function block that displays the stars with an arrow pattern as shown below. The user input is an integer which determines the number of rows and columns in the pattern. Note that in "odd version", a small letter 'v' is used.

If the input integer is 0, negative or an even number, then your program should print the character 'E' (which represents error).

Even Version (even 6th digit in student ID)

<u>Sample display</u>

```
7
*************
*     ^     *
*    ^ ^    *
*   ^ | ^   *
*     |     *
*     |     *
*************
```

Odd Version (odd 6th digit in student ID)

<u>Sample display</u>

```
7
*************
*     |     *
*     |     *
*   v | v   *
*    v v    *
*     v     *
*************
```

<u>More examples</u>

Input	**Output (Even Version)**	**Output (Odd Version)**

5	`*********`
	`*   ^   *`
	`*  ^|^  *`
	`*   |   *`
	`*********`
		`*********`
		`*   |   *`
		`*  v|v  *`
		`*   v   *`
		`*********`

3	`*****`
	`* | *`
	`*****`
		`*****`
		`* | *`
		`*****`

1	`*`	`*`

9	`*****************`
	`*       ^       *`
	`*      ^ ^      *`
	`*     ^   ^     *`
	`*    ^  |  ^    *`
	`*       |       *`
	`*       |       *`
	`*       |       *`
	`*****************`
		`*****************`
		`*       |       *`
		`*       |       *`
		`*       |       *`
		`*    v  |  v    *`
		`*     v   v     *`
		`*      v v      *`
		`*       v       *`
		`*****************`

4	E	E

<u>IMPORTANT NOTE</u>: Compare your program output with the sample display. There should be **<u>NO</u>** prompt messages for the input size and **<u>NO</u>** additional blank lines in the output. **<u>DO NOT add extra text</u>** in the output which will be considered as incorrect in marking.

Submission

(1) The screencast video in mp4 format;
(2) The original transcript file of the screencast video in docx format;
(3) The final source code, which should be named with your student name and ID (12345678A_ChanTaiMan.cpp)

Submit item (1) and (2) through the given MS Form available in the Blackboard assignment submission page, and item (3) to Blackboard directly.

Grading

Part A: Screencast

Your video and the transcription will be checked, both manually and with the help of AI-assisted marking tool, on whether the necessary contents and presentation stated in the requirements are clearly displayed and recorded. There will be marks deduction if the requirements are not met, or any required information was not shown in the screencast video.

Part B & C: Coding

Your program (i.e. the template file with your answers) will be executed by script with different test cases in **Microsoft Visual Studio** using the **Release** setting. The tester will execute the program and enter the question number in "Program Selection Menu" to test a particular question. The program will be restarted for testing each question individually.

You need to follow **EXACTLY** the above input and output requirements. Any deviation from the requirement is considered as incorrect and **no mark** is given for that test case.

Marks deduction

Late submission: 100% deduction. **No late submission is allowed**. Submit your work to Blackboard some time ahead of the deadline. Late submissions due to slow internet speed will not be accepted.

Syntax error: 5% - 20% deduction depends on the seriousness of the syntax error. You will get **0 mark** if your program contains too many syntax errors. Check your final source file using Microsoft Visual Studio (not those online compliers) carefully before submission.

Runtime error: No mark for the particular test case that triggers the runtime error (e.g. infinite loop, divide by zero, etc.).

Logic error (bug): No mark for the particular test case that deviates from the requirement. Note that a logic error may lead to **failure in ALL test cases** of a question, e.g. displaying incorrect messages, incorrect spelling and spacing, incorrect number format, or incorrectly decide the odd/even version, etc.

Unnecessary prompt messages: 15% deduction.

✡✡

Ensure the originality of your work. Plagiarism in any form is highly prohibited.

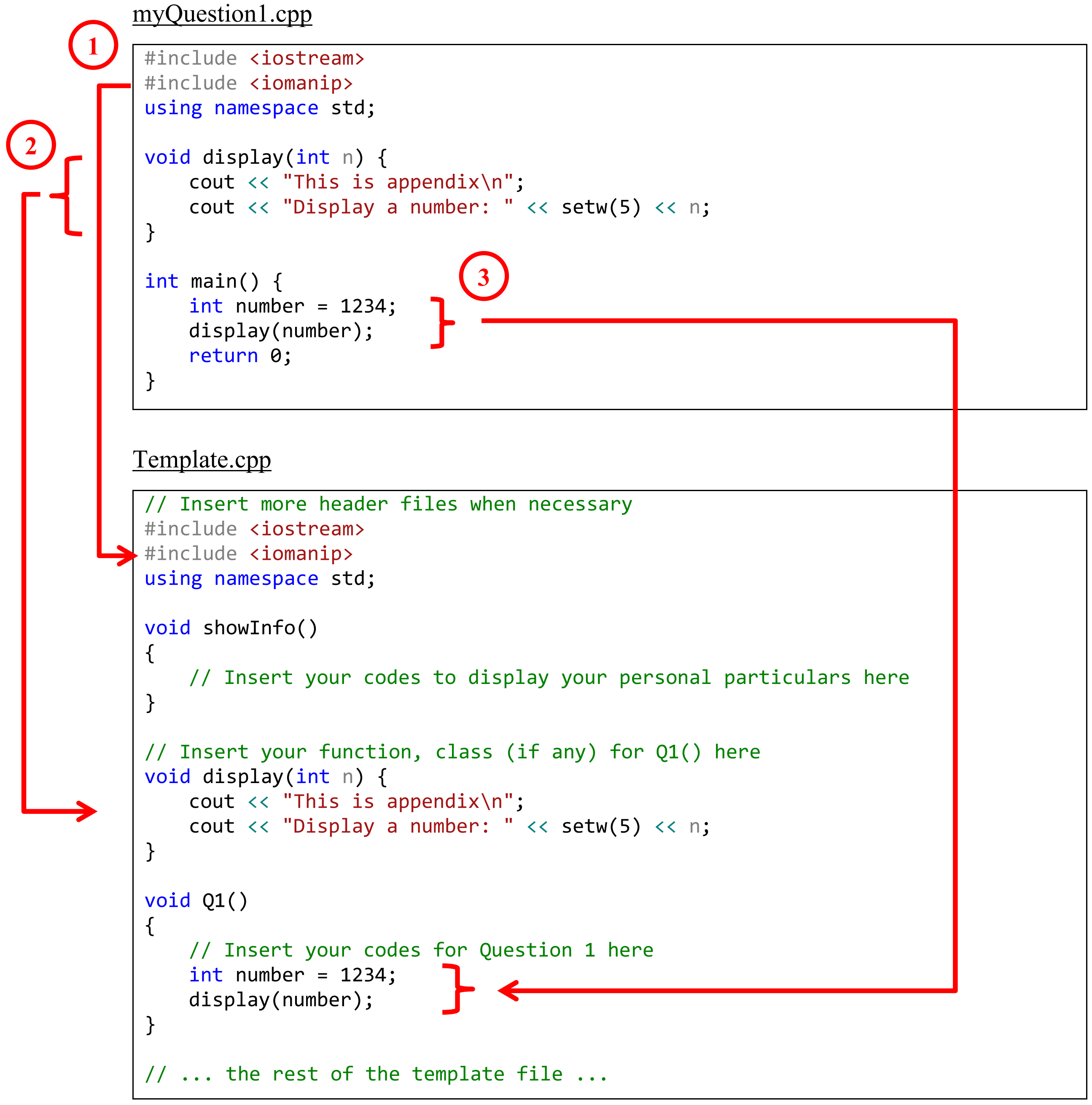
✡✡

Checklist for your use:

- ☐ My source code has been successfully compiled under Microsoft Visual Studio.
- ☐ My screencast video showed my personal profile page of the Blackboard platform.
- ☐ I included my voice in the screencast video. My presentation is in English.
- ☐ My screencast video has been recorded to show the computer screen and enable transcription.
- ☐ In Q2 and Q3, my source code has been tested with different test cases, including my own test cases.
- ☐ There are **NO** prompt messages when my codes in Q2 and Q3 are executed.
- ☐ My personal particulars have been filled in the showInfo function block.
- ☐ I have submitted the video and transcript file correctly via the MS form.
- ☐ My final program submission is in .cpp format with a correct filename.

Additional Information

If you implement the questions in separated source files, you need to copy the program codes into the template file for assignment submission. Make sure to **test the final source file (i.e. template file with your answers)** in Microsoft Visual Studio before submission.



myQuestion1.cpp

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

void display(int n) {
    cout << "This is appendix\n";
    cout << "Display a number: " << setw(5) << n;
}

int main() {
    int number = 1234;
    display(number);
    return 0;
}
```

Template.cpp

```cpp
// Insert more header files when necessary
#include <iostream>
#include <iomanip>
using namespace std;

void showInfo()
{
    // Insert your codes to display your personal particulars here
}

// Insert your function, class (if any) for Q1() here
void display(int n) {
    cout << "This is appendix\n";
    cout << "Display a number: " << setw(5) << n;
}

void Q1()
{
    // Insert your codes for Question 1 here
    int number = 1234;
    display(number);
}

// ... the rest of the template file ...
```

1. The **header files** included in your program should also be included in the template file.
2. The **user-defined function / class** for a question should be copied **before** the question.
3. The program **main body, except "return 0"**, should be copied to the function body of the corresponding question.

- End -