SEHH2042 Computer Programming
Group Project – Student Management System
(Due: 23:59, 27 Apr 2025, Sunday)

Expected Learning Outcomes

- develop computer programs in one or more high level language programming environment;
- design and develop structured and documented computer programs;
- explain the fundamentals of object-oriented programming and apply it in computer program development;
- integrate the computer programming techniques to solve practical problems.

Introduction

In this project, you are going to form group with **5 to 6 members** and develop a "Student Management System" that runs in the command line environment. The system maintains student records of a college. The system allows college administrative staff to check and update academic details of students.

Tasks

- Each group is required to write a Win32 Console Application program called **SMS.cpp**.
- Each student is required to submit a video recording (at most 2-minute long) to demonstrate his/her individual contribution in the group project.
- Each student is required to submit a Peer-to-Peer evaluation form (through the given Word file) via Blackboard.

Program Requirements

R0 When the program starts, the console should display a welcome message, followed by the Main Menu of the program. Users can enter the options of the corresponding actions (see **R1** to **R6** below).

```
Welcome Message designed by your group
*** SMS Main Menu ***
[1] Load Starting Data
[2] Show Student Records
[3] Add/Delete Students
[4] Edit Students
[5] Generate Transcript
[6] Credits and Exit
**********************
Option (1 - 6):
```

R1 [1] Load Starting Data

When the user inputs 1 in the Main Menu, the system is loaded with starting data. The starting data includes student records as shown in R1.1 below. After the starting data is loaded, the system returns to the Main Menu.

R1.1 The starting data to be loaded, together with the required data format of the data fields, are described below.

Student records

Student Name	Student ID	Major	Year	GPA
CHAN Tai Man	S243560	Information Engineering	1	4.00
CHEUNG Jacky	S232210	Civil Engineering	2	2.48
PAN Peter	S222343	Global Business	3	3.42
WONG Kam	S244617	Educational Psychology	1	2.86

Table 1. Student Records

Student records (continued with subject results)

Student Name	Subject Code	Subject Name	Credit	Grade
CHAN Tai Man	ENG2042	Introduction to C++	3	A
	ENG2219	Signal Processing	3	A
	LCH1302	Professional English Writing	2	A
	LCH1019	Japanese I	2	--
CHEUNG Jacky	ENG1113	Structural Fundamental	3	B
	ENG2250	Engineering Mathematics	3	B+
	ENG2041	Applied Computing	3	F
	LCH1302	Professional English Writing	2	A-
	LCH1019	Japanese I	2	B
	BUS1021	Personal Financial Planning	3	--
	ENG2042	Introduction to C++	3	--
PAN Peter	BUS1021	Personal Financial Planning	3	A
	BUS2002	Introduction to Economics	3	A+
	BUS3006	Understanding Globalization	3	B-
	BUS4510	Business Project Management	4	A-
	BUS5523	Final Year Project	5	B+
	LCH1019	Japanese I	2	C
WONG Kam	PSY1234	Introduction to Psychology	2	C
	PSY2233	Sociology	3	B
	PSY2190	Human Behavior	3	B+

Table 2. Student Records (continued with subject results)

Data format of each field:

- **Student Name**: A string consists of surname and given name. Surname should be formatted to all capital letters. The full name is assumed to be at most 30 characters long, including white spaces.
- **Student ID**: A string that **uniquely** identifies a student in the system. It consists of 7 characters, no white space:
  - 1st character: 'S' indicates the record is student.
  - 2nd and 3rd characters: a 2-digit number, indicate the cohort of the student.
    - If a student entered the college in 2023, the cohort number is 23.
  - 4th to 6th characters: a 3-digit number randomly assigned by the system when the student record is added.
  - 7th character: check digit, calculated as the sum of all digits % 10.
    - E.g. PAN Peter's student ID: (2+2+2+3+4) % 10 = 3
- **Major**: A string that represents the major discipline that the student is studying. It is assumed to be at most 30 characters long, including white spaces.
- **Year**: An integer number indicates the year of the student in the college, which can be calculated according to the cohort.
  - Note: Since new academic year begins in Sept 2025, the latest cohort should be 2024. Students admitted in 2024 are Year 1 students.
- **GPA**: A floating point number that represents the grade point average (GPA) of a student, correct to 2 decimal places, which is calculated by the following formula:

$$\text{GPA} = \frac{\sum_{n=1}^{N} \text{Subject Grade Point}_n \times \text{Subject Credit Value}_n}{\sum_{n=1}^{N} \text{Subject Credit Value}_n}$$

  - Note: If a student has not completed any subject, his/her GPA should be displayed as "N/A".
- **Subject Code**: A string that represents the code of a subject. It consists of 3 capital letters ('A' – 'Z') followed by 4 digits (0 – 9). It is 7-character long without white space.
- **Subject Name**: A string that represents the name of a subject. It is assumed to be at most 40 characters long, include white spaces.
- **Credit**: An integer number represents the number of credits of a subject. A subject can be 2 – 5 credits only.
- **Grade**: A string that represents the subject grade student achieved. The range of grades and their corresponding grade points are listed in the following table.

Grade	A+	A	A-	B+	B	B-	C+	C	C-	D+	D	F
Grade Point	4.3	4.0	3.7	3.3	3.0	2.7	2.3	2.0	1.7	1.3	1.0	0

  - Note: If the subject is still in progress, the grade is displayed as "--". Such subject is not included in GPA calculation.

R1.2 Options 2 to 5 in the Main Menu are enabled only after the system is loaded with the starting data. If the user enters options 2 to 5 before starting data is loaded, an error message should be shown, and then the system returns to the Main Menu.

R2 [2] Show Student Records

[After the starting data is loaded] When the user inputs 2 in the Main Menu, the system displays all the fields of all student records as shown in Table 1. A tabular format should be displayed. User can choose to display the records in (1) ascending alphabetical order based on the student names, or (2) descending order based on GPA values.

Data shown should be the **latest** set of data since the starting data is loaded (updates on the student records resulted by the operations under options 3 or 4 in the Main Menu should be included). After showing the records, the system returns to the Main Menu.

R3 [3] Add/Delete Students

[After the starting data is loaded] When the user inputs 3 in the Main Menu, the system prompts for the next user input of a Student ID. If such Student ID does not exist in the system, it is an *add student* operation. Otherwise, it is a *delete student* operation.

You can assume the system will store at most 100 student records.

Add student

The system asks the user to input the required information of a new student: (1) Student Name, (2) Cohort Year, and (3) Major. After getting all user inputs, the system should generate a Student ID and determine the Year, and then the new student should be added into the system. For any invalid inputs (e.g., wrong year input.), the system allows **TWO** more retries. With more than **THREE** times of invalid inputs, no student is added. The system prints an appropriate error message and returns to the Main Menu.

Delete student

The system displays the information of the student (including all the fields in Table 1), and prompts for user's "Yes/No" confirmation on the delete operation. The student record is deleted from the system if it is confirmed.

A message showing the summary of the above operation is then displayed, and the system returns to the Main Menu.

R4 [4] Edit Students

[After the starting data is loaded] When the user inputs 4 in the Main Menu, the system prompts for the user input of a Student ID. If such Student ID does not exist in the system, the system displays an error message and returns to the Main Menu. Otherwise, it displays the Edit Student Menu as shown below and allows further processing on the particular student (e.g., S243560) as follows (see R4.1 to R4.4 below).

```
Action for Student ID: S243560
***** Edit Student Menu *****
[1] Edit Student Name
[2] Edit Major
[3] Edit Subject List
[4] Return to Main Menu
****************************
Option (1 - 4):
```

R4.1 [1] Edit Student Name

When such option is chosen, the system shows the current name of the student and prompts for the user input of a new name. Then the system prompts for user's "Yes/No" confirmation. The student name is updated if it is confirmed. Then the system returns to the Edit Student Menu.

R4.2 [2] Edit Major

When such option is chosen, the system shows the current major of the student and prompts for the user input of a new major. Then the system prompts for user's "Yes/No" confirmation. The student major is updated if it is confirmed. Then the system returns to the Edit Student Menu.

R4.3 [3] Edit Subject List

When such option is chosen, the system allows users to make changes to the subject list and subject grade of the student. The system first prompts for the user input of a subject code. If the format of subject code is correct, the user is prompted for further inputs according to the following situations:

(1) The subject code exists in the student subject list
The system prompts for user input of a new subject grade. Note that any subject with existing letter grade cannot be changed to "--" (in progress).

(2) The subject code exists in the system subject list, but not in the student subject list
The system adds the subject to the student subject list, and prompts for user input of a subject grade. The user can input "--" to indicate that the subject is in progress.

(3) The subject code is new to the system
The system prompts for the user input of the name and number of credits of the subject. Then the subject should be added to the system subject list and the student subject list. The system further prompts for user input of a subject grade. The user can input "--" to indicate that the subject is in progress.

After the above process, the system updates the GPA of the student according to the latest subject list. For any invalid input, the system allows **TWO** more retries. With more than **THREE** rounds of invalid inputs, the system prints an appropriate error message and returns to the Edit Student Menu.

R4.4 [4] Return to Main Menu

When such option is chosen, the system returns to the Main Menu.

R4.5 Stay at the Edit Student Menu

Following R4.1, R4.2 and R4.3, after an operation for a student, the system should display the **change** (the original and the new values) in the student record by that operation, and then stays at the Edit Student Menu.

R5 [5] Generate Transcript

[After the starting data is loaded] When the user inputs 5 in the Main Menu, the system prompts for the user input of a Student ID. If such Student ID does not exist in the system, the system displays an error message and returns to the Main Menu. Otherwise, it displays the transcript of the student (e.g. S232210) as shown below. The list of subjects is displayed in **ascending order** based on subject code. After the subject list, the number of credits attained (i.e., of passed subjects) and the GPA are displayed.

```
Name      : CHEUNG Jacky
Student ID: S232210
Major     : Civil Engineering
Year      : 2

Subject                                              Credit  Grade
-------------------------------------------------------------------
BUS1021  Personal Financial Planning                 3       --
```

```
ENG1113  Structural Fundamental                    3       B
ENG2041  Applied Computing                         3       F
ENG2042  Introduction to C++                       3       --
ENG2250  Engineering Mathematics                   3       B+
LCH1019  Japanese I                                2       B
LCH1302  Professional English Writing              2       A-
-----------------------------------------------------------------
Credits attained: 10
GPA: 2.48
```

After displaying the transcript, the system returns to the Main Menu.

R6 **[6] Credits and Exit**

When the user inputs this option, the system prompts for user's "Yes/No" confirmation. If the user confirms, the system displays the personal particulars (student name, student ID, tutorial group) of the group members and terminates. Otherwise, the system returns to the Main Menu.

R7 For all Yes/No confirmation, the user should input 'Y' or 'y' for Yes and 'N' or 'n' for No. Other input is not acceptable and the system should ask the user to confirm again.

R8 Suitable checking on user's input is expected, except in situations with assumptions stated in the requirements above. Appropriate error messages should be printed whenever unexpected situation happens, e.g., input value out of range, incorrect character input, etc.

R9 The use of functions (in addition to main function) and classes (i.e., OOP design) are expected in your program. Appropriate comments should be added in your source code file.

R10 Creativity and Critical Thinking: Use suitable format to present all instructions and information clearly and neatly. Additional useful features can be added to the system.

Tips

1. To handle unexpected input error (e.g. input a character to an integer variable), you may use the following code appropriately in your program:

```
cin.ignore();    // Discard the content in the input sequence.
cin.clear();     // Reset the input error status to no error.
```

Video Requirements

This is an **individual task** under this group project. Each student needs to create a video recording which records either (1) your explanation on the working algorithms of the codes that you designed

and wrote, or (2) the testing of project codes (those you wrote or the whole group's work) using test case scenarios. See points below for the specific requirements of the video recording:

- The duration of the video should be between 1 to 2 minutes long. Use **MS Teams** to record. Marks will be deducted if such requirement is not met.
- The video recording is used to demonstrate your contribution to the group project. If your work done is too much to be included in the 2-minute video, choose the most important/ representative work to record and explain.
- At the beginning, **clearly mention** the objective of your video: whether you are going to (1) explain the working algorithms of your codes, or (2) run and test your project codes. An example starting dialog could be *"The objective of this video is to ……"*.
- The video should include your voice recording (in English) as you give the presentation (either code explanation or the code testing). Your voice should be clear and loud enough.
- The video should show the computer screen as you give the presentation. Suitable cursor movement or text highlighting by mouse action should be present.
- In addition to your name shown by MS Teams, you should **also** show your English name, Student ID and lecture group on the screen (e.g., put down the information using Notepad and display it). The video should keep showing it for identity verification. It is optional to show your face in the recording.
- Show the whole screen, not just part of the screen. While showing the source codes or code-testing results, make sure the text is clearly shown, large enough and visible.
- The computer may contain multiple windows to show the output and the relevant program code at the same time.

List of suggested dialogues

The table below provides a list of useful dialogues to guide your presentation during the video recording. You are suggested to make good use of them to deliver a clear and easy-to-understand presentation in the video.

You may say …	**Purpose**
"The objective of this video is to …"	To state your objective of the video at the beginning.
"I am going to test the XXX function in the program. First, I enter …" Avoid saying things like: *"I am going to test the R2 function in the program. First, I enter …"*	To indicate you are going to test a particular function in the program. The function to test could be: • Show Student Records • Add or Delete Student • Edit Student • Edit Major • Edit Subject List • Generate Transcript

"I am going to test how the invalid inputs are handled. First, I enter …"	To indicate you are going to test the program with some invalid inputs.
"I am going to explain the working algorithm of the XXX function in the program. First, in the code, …" Avoid saying things like: *"I am going to explain the working algorithm of the R2 function in the program. First, in the code, …"*	To indicate you are going to explain the working algorithm of a particular function in the program. The function to explain could be: • Show Student Records • Add or Delete Student • Edit Student • Edit Major • Edit Subject List • Generate Transcript

Submission

Source File: Each group submits one source code file (i.e., **SMS.cpp**).

Video Recording: Each student submits the shared video link via Blackboard, through which your video recording can be viewed by your subject lecturer successfully.
[IMPORTANT: Remember to set up the access right correctly for the shared link. You can refer to the guidelines available in Blackboard on how to prepare the recording and set up the shared link.]

Peer-to-peer Evaluation: Each student fills in the peer-to-peer evaluation form (MS Word) and submits via Blackboard.

All submission should be done **through Blackboard by 23:59, 27 Apr 2025 (Sunday)**. Late submission is subject to 20% deduction in your final marks for each day. No late submission is allowed **4** days after the due date.

Components Weighting

1. Program (SMS.cpp)	80%	(Group & Individual)*
2. Video Recording	20%	(Individual)
Total	100%	

* Marks for 1. Program (SMS.cpp) is determined by the group-based marks (80%) and percentage of individual contribution (20%), where the percentage of individual contribution is directly proportion to the average marks given by group members in the peer-to-peer evaluation.

Grading Criteria (Program)

Your program will be executed with different test cases in **Microsoft Visual Studio**. Any deviation from the requirement is considered incorrect and **no mark** is given for that case. Your program will also be marked based on its user-friendliness and creativity (e.g., information display, appropriate prompt messages and action result messages if needed).

Aspects	Percentage
Program correctness (Follow ALL instructions, marks deduction on errors found)	70%
Program design (Appropriate use of functions, use of class, modularity, etc.)	5%
Program standard (Use of variable names, indentation, line spacing, clarity, comments, etc.)	5%
Algorithm design (Use of reasonable algorithms and data structures)	5%
User-friendliness (Clear guidelines to users, messages to users, etc.)	5%
Creativity and critical thinking (Additional useful features)	10%
Total (Group Mark)	**100% (max)**

Note: the length of your program does not affect the grading of the assignment. However, appropriate use of loops and functions are expected to avoid too many repeated codes in your program, which contributes to the program design score of this assignment.

Marks Deduction (Program)

Syntax error: 100% deduction. You will get **0 mark** if your program fails to be compiled.
Runtime error: No mark for the particular test case that triggers the runtime error.
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 one particular set of requirements.

Grading Criteria (Video Recording)

Aspects	Percentage
Basic requirements (video duration, language used, showing identity, stating objective, etc.)	25%
Presentation contents (suitable content, logical presentation, etc.)	50%
Communication clarity (text display, voice, mouse actions, etc.)	25%
Total	**100% (max)**

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

- End -