

**ITS41804 Problem Solving and Programming Design**

**Group Project**

**Cover Sheet**

**HAND OUT DATE: 16 October 2024**

**HAND IN DATE: 25 November 2024**



**WEIGHTAGE: 35%**

**SEMESTER: AUGUST 2024**

**Instructions to student:**

* This is a Group Project. You can form a group of 3-4 persons.
* Complete this cover sheet and attach it to your group project – this should be your first page!
* One group only need to submit once. Refer to submission guideline for detail.

|  |  |
| --- | --- |
| **Student declaration:** | |
| ***I declare that:***   1. ***I understand what is meant by plagiarism*** 2. ***I understand that the use of Generative AI (eg: ChatGPT) is not permitted for this assignment*** 3. ***The implication of plagiarism has been explained to me by our lecturer***   ***This assignment is my own work.*** | |
| **Name and Signature** | **Student ID** |
| **BRYAN LAM WEN JIA**  **MATTHEW LIM YI HONG**  **NG JING HAN**  **TEE HAO XIANG** | **0377367**  **0375947**  **0378821**  **0378210** |

## Scenario:

As part of a group project, you and your teammates are tasked with developing a Student Management System for a school. This system will have multiple features including managing student records, calculating grades, handling large datasets with iteration, and storing data in text files. Additionally, the system will integrate functions to search records, provide statistical analysis, and produce a report of student performance.

Each member of the group will be responsible for different modules of the system. The project must include efficient handling of arrays, strings, file I/O operations, and iteration across multiple datasets.

**Functional Requirements:**

1. **Student Data Input and Storage:**

* Develop functionality to add student records.
* Store student information in a text file named students\_data.txt. This file will include:

First Name, Last Name, Student ID, Grades for five subjects (Math, English, Science, History, Art)

Example line in students\_data.txt: ***John Doe 12345 78 85 90 88 76***

1. **Processing and Calculations:**

* Read from the students\_data.txt file for processing.
* Implement iteration to calculate each student’s average grade and determine the highest and lowest performing student.
* Calculate and output overall class performance statistics (e.g., average grade across all students, highest and lowest scores).

1. **Search Functions:**

* Implement a search function where users can search for a student by their last name or student ID.
* Implement error handling for invalid searches.

1. **Statistical Reports:**

* Generate a report that summarizes:
* The top 3 students based on their average grades.
* Students who are at risk (with average grades below 50).
* Subject-wise highest and lowest scores.
* Output the report both to the screen and save it to a file named report.txt.

1. **Interface**

You are required to create an interface to integrate all the functionality of the system for the users to interact with your system.

**Example of screenshot of each function:**

1. Add Student Record:

Enter First Name: Alice

Enter Last Name: Johnson

Enter Student ID: 54321

Enter grades (Math, English, Science, History, Art): 85 88 90 92 80

Student record added successfully!

1. Search Student Record:

Enter Last Name to search: Johnson

Record found:

Alice Johnson (ID: 54321) - Grades: 85 88 90 92 80

1. Statistical Report (report.txt)

Class Performance Report:

-------------------------

Top 3 Students:

1. Alice Johnson (ID: 54321) - Average Grade: 88.0

2. John Doe (ID: 12345) - Average Grade: 83.4

3. Bob Smith (ID: 67890) - Average Grade: 81.5

Students At Risk (Average Grade < 50):

None

Subject-Wise Highest and Lowest Scores:

Math - Highest: 90 (Alice Johnson), Lowest: 70 (Bob Johnson)

English - Highest: 92 (Alice Johnson), Lowest: 75 (Bob Johnson)

...

**Submission guideline**

**Submission platform: MyTIMES**

Pseudocode, Flowchart, and C Program Submission: (by group leader only)

1. Cover page which shows your names, student ID, and signature. Please include Assessment Criteria form as well.
2. Submit your program source code in .c format. Pseudocode and Flowchart in Word Document

Peer Assessment Review Submission: (everyone)

Fill in the online form to evaluate: yourself and your group member’s contribution in this group project. One assessment form is for one person only. The assessment link will be provided.

**Assessment Criteria (Group Project) – 25%**

|  |  |  |
| --- | --- | --- |
| **Task** | **Maximum marks** | **Awarded marks** |
| **Pseudocode and Flowchart** | 20 |  |
| **Program Flow/ Menu**  Correct programming logic and proper programming design  There are no run-time errors for the program.  Menu can function well. | 10 |  |
| **Add Student Record** | 15 |  |
| **Search Function** | 20 |  |
| **Student Performance** | 25 |  |
| **Other requirements**  Proper code indentation.  Comment line is available | 10 |  |
| **Program not be able to compile or execute -50** | | |
| **Total** | **100** |  |

**Assessment Rubric**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | Outstanding (8-10) | Mastering (6-7) | Developing (4-5) | Beginning (0-3) |
| Apply different programming logic and problem-solving skills in the application development. | Solution presented solves the problem stated correctly and meets all requirements of the problem according to the application theme.  Solution of the functionalities are clearly presented. Solution represents an elegant and effective way to solve the problem and is not overly complicated than is necessary.  Solution is clearly adhered to the programming style. | Solution is mostly correct, satisfying most of the above criteria under the exemplary category, but contains some minor pitfalls, errors/flaws or limitations. | Solution demonstrates a viable approach toward solving the problem but contains some major pitfalls, errors/flaws or limitations. | No solution provided or critical elements of the solution are missing or significantly flawed.  Solution does not demonstrate sufficient understanding of the problem and/or any reasonable directions to solve the problem.  Solution does not adhere to the programming style. |

**Pseudocode**

Define maxstudent as 100

Define firstname[maxstudent][50] to save the first name

Define lastname[maxstudent][50] to save the last name

Define studentid[maxstudent][50]to save the student id of student

Define grade[maxstudent][5] to save the grades for Math, English, Science, History and Art

Define studentcount to save the number of students and initialise to 0

Define average[maxstudent] to store the average grades for each student

Declare function addstudent() to get student records

Declare function searchstudents() to search for student records

Declare function report() to create a statistical report

Begin

Read input into variable “a”

Print “Student Management System”

Print “1) Add student record”

Print “2) Search student record”

Print “3) Statistical report”

Print “4) Exit”

Prompt user to “Select an option:”

Enter a

if a=1

Run ‘addstudent()’

if a=2

Run ‘searchstudents()’

if a=3

Run ‘report()’

if a=4

Print “Exited”

end\_if

End

Function addstudent()

Begin

Print “Enter First Name:”

Enter ‘firstname[studentcount]’

Print “Last First Name:”

Enter ‘lastname[studentcount]’

Print “Enter Student ID:”

Enter ‘studentid[studentcount]’

Print “Enter grade (Math, English, Science, History, Art):”

for each subject from Math to Art

Enter ‘grade[studentcount]’

end\_for

Open file “student\_data.txt” in append mode

if file opened successfully

Write ‘firstname’ ‘lastname’ ‘studentid’ ‘grade[studentcount][0]’ ‘grade[studentcount][1]’ ‘grade[studentcount][2]’ ‘grade[studentcount][3]’ ‘grade[studentcount][4] to the file

Close the file

Print “Student record added successfully”

Increment the ‘studentcount’

else

Print “Not found the file”

end\_if

Return to main function

End

Function searchstudent()

Begin

Open file “student\_data.txt” in read mode

if file does not open successfully

Print “Error: Could not open file.”

if file open successfully

Print “Search by Last Name (1) or Student ID (2): ”

Define ‘choice’ as integer

Enter ‘choice’

if choice=1

Begin

Define lastName[50] as string

Define ‘found’ to track if any last name is found and initialise to 0

Print “Enter Last Name: ”

Enter ‘lastName’

for each student record ‘i’ from 0 to maxstudent -1

Read ‘firstname[i]’ ‘lastname[i]’ ‘studentid[i]’ ‘grade[i][0]’ ‘grade[i][1]’ ‘grade[i][2]’ ‘grade[i][3]’ ‘grade[i][4]’ from the file

if ‘i’ not equals to 8

return 0

if lastname[i]=lastName

Print “Record found: ‘firstname[i]’ ‘lastname[i]’ (ID-‘studentid[i]’) – Grades: ‘grade[i][0]’ ‘grade[i][1]’ ‘grade[i][2]’ ‘grade[i][3]’ ‘grade[i][4]’

Set ‘found’=1

end\_if

end\_for

if ‘found’=0

Print “No records found for last name: ‘lastName’ ”

end\_if

end\_if

else if choice=2

Begin

Define studentID[50] as string

Print “Enter Student ID: ”

Enter ‘studentID’

Open file “student\_data.txt” in read mode

while read ‘firstname[studentcount]’ ‘lastname[studentcount]’ ‘studentid[studentcount]’ ‘grade[studentcount][0]’ ‘grade[studentcount][1]’ ‘grade[studentcount][2]’ ‘grade[studentcount][3]’ ‘grade[studentcount][4]’ from the file

if there are no 8 values read

Break

end\_if

if studentid[studentcount]=studentID

Print “Record found: ‘firstname[studentcount]’ ‘lastname[studentcount]’ (ID-‘studentid[studentcount]’) – Grades: ‘grade[studentcount][0]’ ‘grade[studentcount][1]’ ‘grade[studentcount][2]’ ‘grade[studentcount][3]’ ‘grade[studentcount][4]’”

Close the file

end\_if

end\_while

Print “No records found for Student ID: ‘studentID’”

Close the file

End

Function report()

Begin

Open file "student\_data.txt" in read mode

Open file "report.txt" in write mode

Define ‘top3[3]’ as float and initialise to 0 to store top 3 students averages

Define ‘top3\_ids[3][10]’ as integer and initialise to 0 to store IDs of top 3 students

Define ‘top3\_firstnames[3][50]’ as string to store first names of top 3 students

Define ‘top3\_lastnames[3][50]’ as string to store last names of top 3 students

Define ‘total\_students’ as integer and initialise to 0

Define ‘at\_risk\_count’ as integer and initialise to 0

Define ‘class\_sum’ as float and initialise to 0

Define ‘subject\_high[5]’ as integer initialise to 0 to store highest grades for each subject

Define ‘subject\_low[5]’ as integer initialise to 100 to store lowest grades for each subject

Define ‘high\_scorer\_first[5][50]’ as string to store first names of highest scorers for each subject

Define ‘high\_scorer\_last[5][50]’ as string to store last names of highest scorers for subject

Define ‘low\_scorer\_first[5][50]’ as string to store first names of lowest scorers for each subject

Define ‘low\_scorer\_last[5][50]’ as string to store last names of lowest scorers for each subject

Define ‘temp\_first[50]’ as a string to store the first names of student

Define ‘temp\_last[50]’ as a string to store the last names of student

Define ‘temp\_id[10]’ as an integer to store the IDs of student

Define ‘grades[5]’ as an array to hold the grades for all 5 subjects.

Define ‘at\_risk\_names[100][50]’ as string to store names of at-risk students

Define ‘at\_risk\_ids[100][10]’ as integer to store IDs of at-risk students

while reading data first name, last name, student ID, and 5 grades from the file

if 8 values read successfully

float ‘avg’ = (grades[0] + grades[1] + grades[2] + grades[3] + grades[4]) / 5.0

‘class\_sum’ = ‘class\_sum’ + ‘avg’

Increment the ‘total\_students’

end\_if

end\_while

for ‘i’ from 0 to 2

Begin

If ‘avg’ > ‘top3[i]’

Begin

For ‘j’ from 2 down to ‘i’+1

Begin

Update ‘top3[j-1]’ to ‘top3[j]’

Update ‘top3\_ids[j-1]’ to ‘top3\_ids[j]’

Copy the first name of the student rank ‘top3\_firstnames[j-1]’ to ‘top3\_firstnames[j]’

Copy the last name of the student rank ‘top3\_lastnames[j-1]’ to ‘top3\_lastnames[j]’

End

Update ‘avg’ to ‘top3[i]’

Update ‘temp\_id’ to ‘top3\_ids[i]’

Copy the current student’s first name ‘temp\_first’ to ‘top3\_firstnames[i]’

Copy the current student’s last name ‘temp\_last’ to ‘top3\_lastnames[i]’

Break

End

end\_if

End

if avg < 50

Begin

Copy the first name ‘temp\_first’ into the ‘at\_risk\_names[at\_risk\_count]’ array

Append a ‘" "’ to the name

Append the last name ‘temp\_last’ to complete the full name

Save the student ID ‘temp\_id’ into the ‘at\_risk\_ids[at\_risk\_count]’ array

Increment the ‘at\_risk\_count’

End

end\_if

for each subject ‘I’ from 0 to 4

if ‘grades[i]’ > ‘subject\_high[i]’

Begin

Update the highest score for subject ‘subject\_high[i]’ to ‘grades[i]’

Copy the current student's first name ‘temp\_first’ into ‘high\_scorer\_first[i]’

Copy the current student's last name ‘temp\_last’ into ‘high\_scorer\_last[i]’

End

end\_if

if ‘grades[i]’ < ‘subject\_low[i]’

Begin

Update the lowest score for subject ‘subject\_low[i]’ to ‘grades[i]’

Copy the current student's first name ‘temp\_first’ into ‘low\_scorer\_first[i]’

Copy the current student's last name ‘temp\_last’ into ‘low\_scorer\_last[i]’

End

end\_if

end\_for

Write “Class Performance Report:” to report file

Write “----------------------------" to report file

Write “Top 3 Students: ” to report file

for each student ‘I’ from 0 to 2 and ‘top3[i]’ > 0

Write the student's rank, first name, last name, student ID, and average grade to report file

end\_for

Write “Students At Risk (Average Grade < 50): ” to report file

if ‘at\_risk\_count’ > 0

for each ‘at\_risk student’ ‘I’ from 0 to ‘at\_risk\_count’-1

Write ‘at\_risk\_names[i]’ and ‘at\_risk\_ids[i]’ to report file

end\_for

else

Write “None” to report file

end\_if

Write “Subject-Wise Highest and Lowest Scores: ” to report file

Define constant ‘subjects[]’ as an array: {“Math”, “English”, “Science”, “History”, “Art”}

for each subject ‘I’ from 0 to 4

Write ‘subjects[i]’ – Highest: ‘subject\_high[i]’ ( ‘high\_scorer\_first[i]’, ‘high\_scorer\_last[i]’), Lowest: ‘subject\_low[i]’ (‘low\_scorer\_first[i]’, ‘low\_scorer\_last[i]’) to report file

end\_for

Calculate ‘overall class average’ as (‘class\_sum’ / ‘total\_students’)

Write “Overall Class Average: ‘overall class average’ ” to report file

Close file “student\_data.txt”

Close file “report.txt”

Write “Report generated successfully in ‘report.txt’.”

End

**Flowchart**

A diagram of a flowchart

Description automatically generated