

# Course Work

**ITS1010 – Programming Fundamentals**

BSc (Hons.) in Computer Science via GDSE



Take-home assignment and VIVA

Total Marks: 100

### Objectives

- Solve fundamental programming problems using a variety of skills and strategies.
- Apply appropriate techniques to create entry-level programs from models.

### Coursework Requirements and Instructions

- You have to focus on how to implement this system using your knowledge of Arrays.
- You must create the application in this coursework as a CLI (Command Line Interface) software.
- You are required to successfully attempt all the parts and face the viva-voce at the end of this coursework.
- Refer to the Coursework Guidelines at the end to understand the specific guidelines to be followed when developing the project required.
- Demo Videos are given at relevant places for you to understand the coursework requirement better.
- Apart from explaining the code you have written, you should be able to explain the theory concepts tested in this coursework.

### Submission

- You should submit the coursework on or before the due date specified.
- Your work is to be made into a .zip file with the file name format **[GDSEBatch\_ITS1011\_Name]**, and submit your deliverables and application code to Google Classroom on or before the deadline.
- E.g., If your GDSE batch is 56 and your name is Nimal Perera,
  - The .zip file name: GDSE56\_ITS1011\_NimalPerera.zip

## Case Study

At IJSE, all the GDSE students have a unique student id. Imagine that in the 1st semester of GDSE, they are supposed to take two mandatory modules, which are Programming Fundamentals and Database Management Systems. Students are supposed to face exams for these modules at the end of the semester.

The instructors and lecturers require a system to manage student marks. As you are a new GDSE student, they have thought to give you a chance of making a system for them.



Figure 1 – Use case Diagram

## Requirement

You are supposed to create a Java application to manage these students' marks. In the application, you need to implement the following use cases.

When you run the application, you should come up with something similar to the following Command-Line Interface (CLI), where the user can enter an option number that he wants to execute. This will be the Home Page of the application that you will be developing.

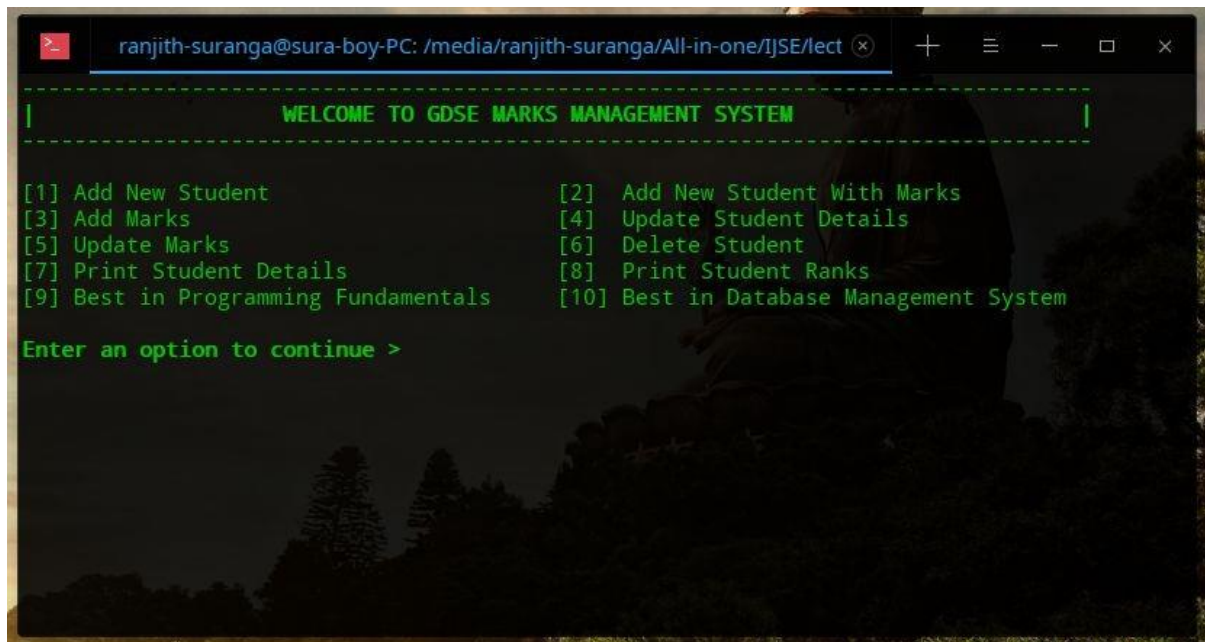


Figure 2 – Home Page of the GDSE Marks Management System

### 1. Add New Student [\[Demo\]](#)

Adding a new student is easy. First, the user needs to enter the Student ID then Student Name. This is only for adding a new student, not for assigning marks. Once the user has added a new student successfully, a message should prompt to ask whether the user wants to add a new student again or to go back to the main menu.

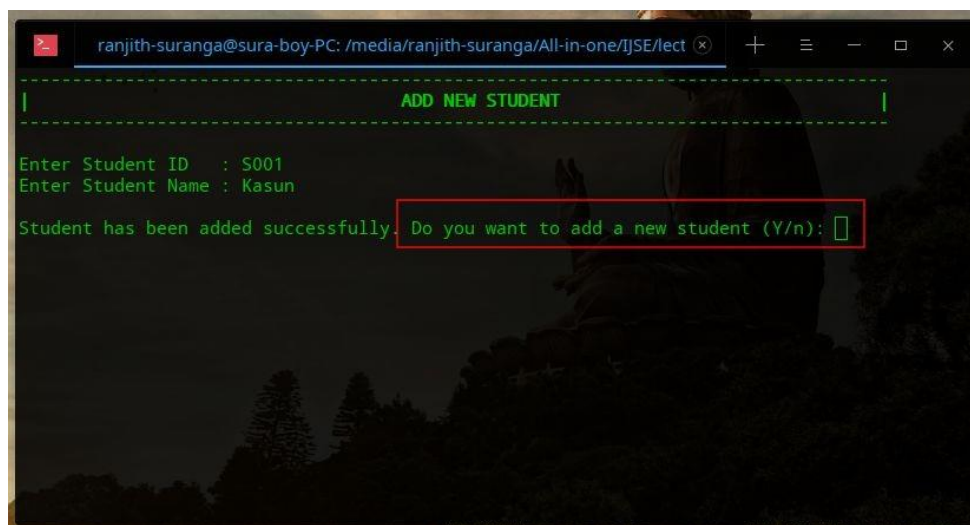
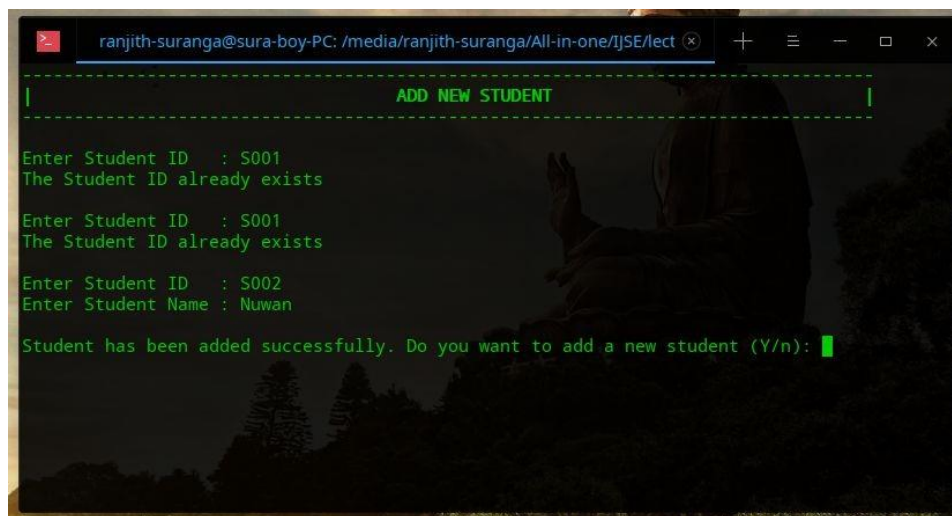


Figure 3 – Add New Student (Successful)

It is not allowed to add the same Student ID again. If the user has entered a student id that already exists, the user should be notified about it and asked to enter the correct Student ID again. This should keep prompting until the user enters a valid Student ID.

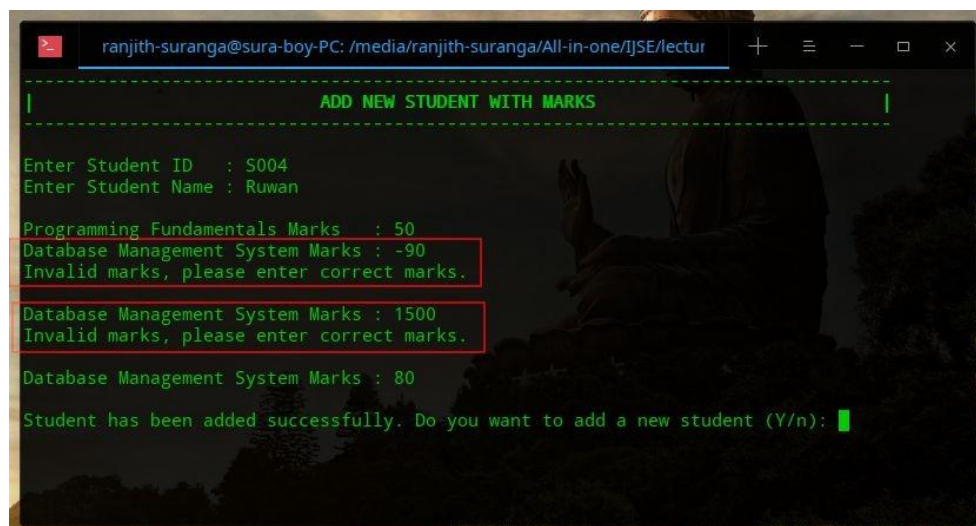


```
ranjith-suranga@suraboy-PC: /media/ranjith-suranga/All-in-one/TJSE/lect
|----- ADD NEW STUDENT -----|
Enter Student ID : S001
The Student ID already exists
Enter Student ID : S001
The Student ID already exists
Enter Student ID : S002
Enter Student Name : Nuwan
Student has been added successfully. Do you want to add a new student (Y/n):
```

Figure 4 – Add New Student (Warning – already exist)

## 2. Add New Student With Marks [\[Demo\]](#)

Previously the user can only add a new student, but with this option, not just the user can add a new student but marks too. When adding marks (assuming the user is only going to enter integer values), marks should be validated, which means marks should be within the range of 0-100. If the user has entered invalid marks (as an example, a negative number or a number greater than 100), the user should be kept prompting until he enters valid marks. Other than that, everything else is the same as above.

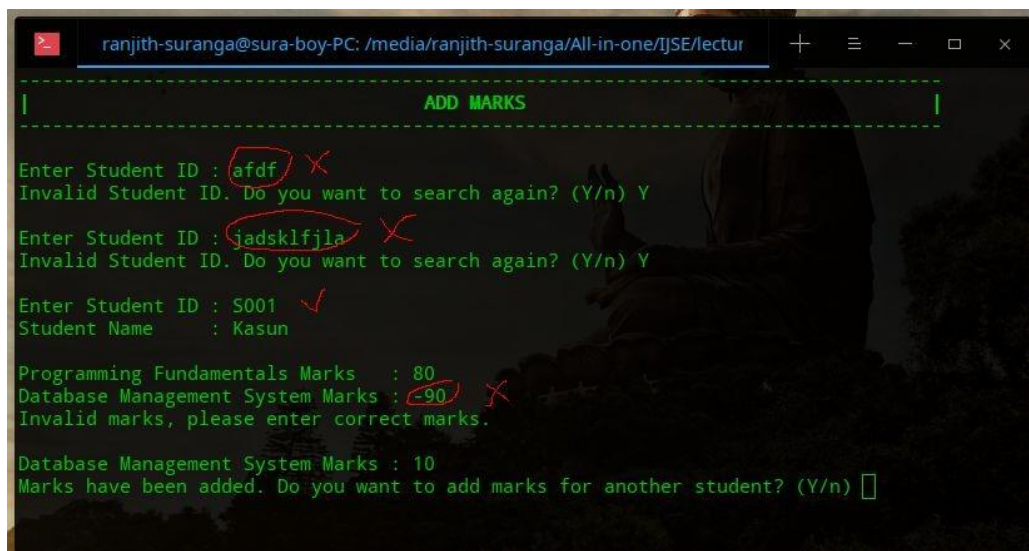


```
ranjith-suranga@suraboy-PC: /media/ranjith-suranga/All-in-one/TJSE/lectur
|----- ADD NEW STUDENT WITH MARKS -----|
Enter Student ID : S004
Enter Student Name : Ruwan
Programming Fundamentals Marks : 50
Database Management System Marks : -90
Invalid marks, please enter correct marks.
Database Management System Marks : 1500
Invalid marks, please enter correct marks.
Database Management System Marks : 80
Student has been added successfully. Do you want to add a new student (Y/n):
```

Figure 5 – Add New Student with Marks

### 3. Add Marks [\[Demo\]](#)

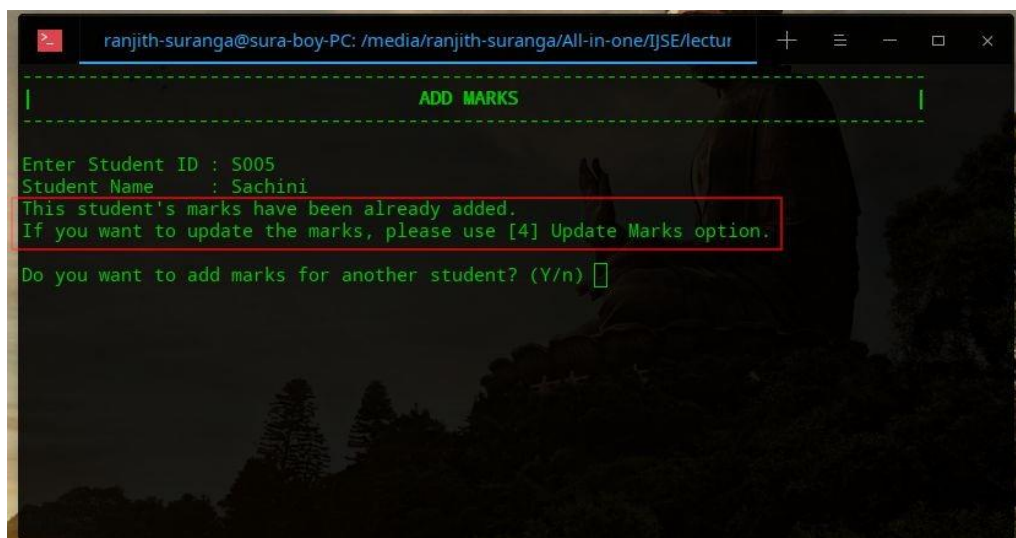
If the user previously has entered student details only, then adding marks of these students can be done with this functionality. First, the user needs to find the student via Student ID. If the user has entered an invalid Student ID, then it should be notified to the user and asked whether to continue search or not. Until the user enters a correct student id, this should keep happening. Once the user has entered a valid Student ID, it should display the Student Name and ask for new marks. Again here, marks should be validated like previously. Once the correct marks have been entered, it should prompt whether to add marks for another student or go back to the main menu.



```
ranjith-suranga@sura-boy-PC: /media/ranjith-suranga/All-in-one/IJSE/lectur
|----- ADD MARKS -----|
Enter Student ID : afd ☒
Invalid Student ID. Do you want to search again? (Y/n) Y
Enter Student ID : jadsklfjla ☒
Invalid Student ID. Do you want to search again? (Y/n) Y
Enter Student ID : S001 ☒
Student Name      : Kasun
Programming Fundamentals Marks : 80
Database Management System Marks : -90 ☒
Invalid marks, please enter correct marks.
Database Management System Marks : 10
Marks have been added. Do you want to add marks for another student? (Y/n) ☐
```

Figure 6 – Add Marks (with validation - 1)

If the user tries to enter marks for a student who has been assigned marks previously, it should be notified as well. This option is only available for the students whose marks still haven't been given.



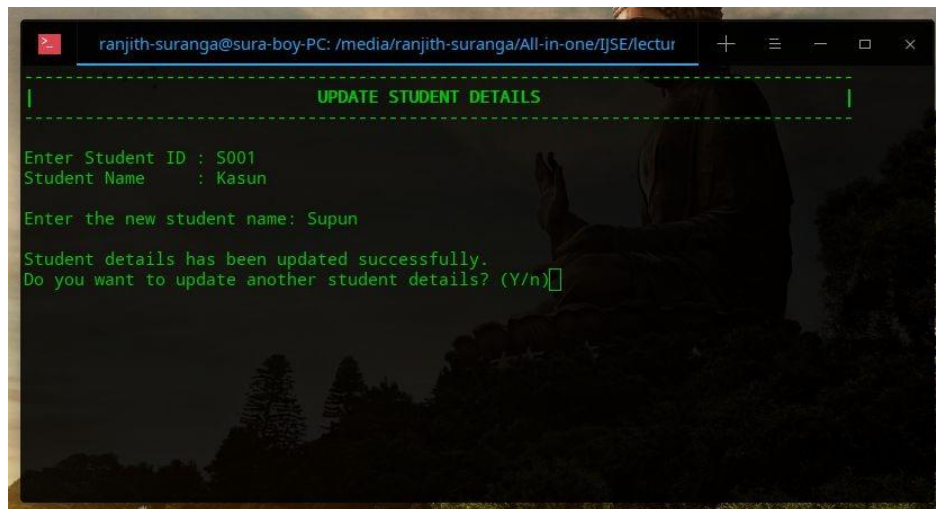
```
ranjith-suranga@sura-boy-PC: /media/ranjith-suranga/All-in-one/IJSE/lectur
|----- ADD MARKS -----|
Enter Student ID : S005
Student Name      : Sachini
This student's marks have been already added.
If you want to update the marks, please use [4] Update Marks option.
Do you want to add marks for another student? (Y/n) ☐
```

Figure 7 – Add Marks (with validation - 2)



#### 4. Update Student Details [\[Demo\]](#)

With this option, the user can update existing student details (Student Name). First, the user should enter a valid Student ID. Otherwise, it should handle like previously. Once the user has entered a valid Student ID, it will display the current Student name and ask for the new student name. Once the data has been updated successfully, it should prompt whether to update another student's details or go back to the main menu.

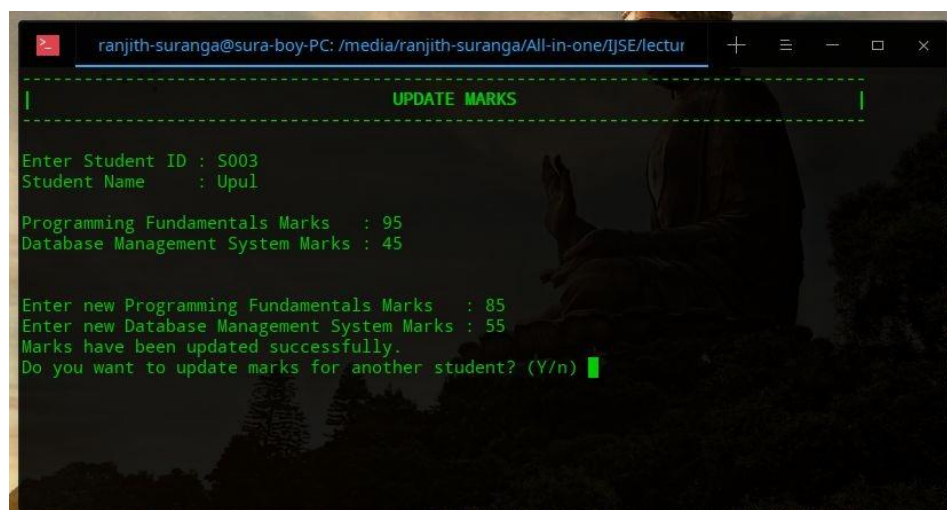


```
ranjith-suranga@sura-boy-PC: /media/ranjith-suranga/All-in-one/IJSE/lectur
|-----UPDATE STUDENT DETAILS-----|
Enter Student ID : S001
Student Name      : Kasun
Enter the new student name: Supun
Student details has been updated successfully.
Do you want to update another student details? (Y/n)
```

Figure 8 – Update Student Details

#### 5. Update Marks [\[Demo\]](#)

With this, the user can update previously added marks. First, the user needs to find the student whose marks should be updated via Student ID. (Invalid Student ID's should be handled like previously) Once the user has entered a valid student ID, it is going to display the current details, which are the student's name and the marks that have been added previously. Then it prompts for new marks. Again, marks should be validated here. Once the update has been done successfully, it should prompt whether to continue updating or go back to the main menu.



```
ranjith-suranga@sura-boy-PC: /media/ranjith-suranga/All-in-one/IJSE/lectur
|-----UPDATE MARKS-----|
Enter Student ID : S003
Student Name      : Upul
Programming Fundamentals Marks : 95
Database Management System Marks : 45
Enter new Programming Fundamentals Marks : 85
Enter new Database Management System Marks : 55
Marks have been updated successfully.
Do you want to update marks for another student? (Y/n)
```

Figure 9 – Update Marks

If the user has entered a valid student id of a student whose marks still haven't been added to the system, then it should be notified as well like follows.

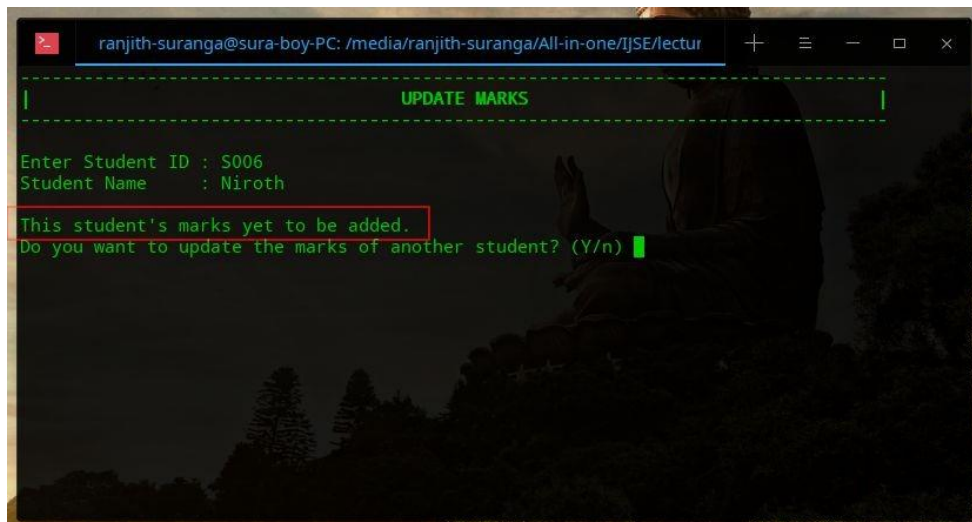


Figure 10 – Update Marks (validation)

## 6. Delete Student

With this option, a student can be deleted from the system. First, the user needs to enter the valid Student ID; otherwise, it should handle like previously. Once the user has entered a valid student id, the student can be deleted. Upon the successful deletion, the user should be prompted whether to delete another student or go back to the main menu.

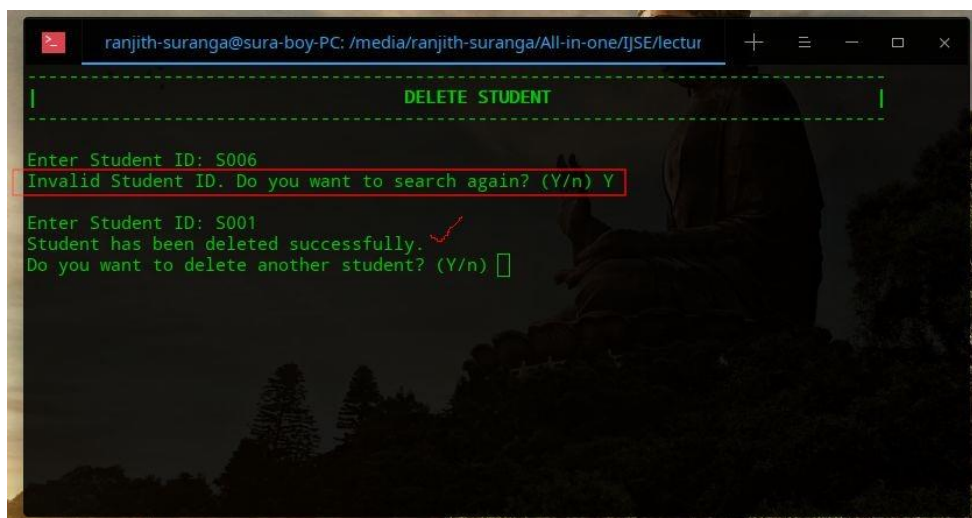
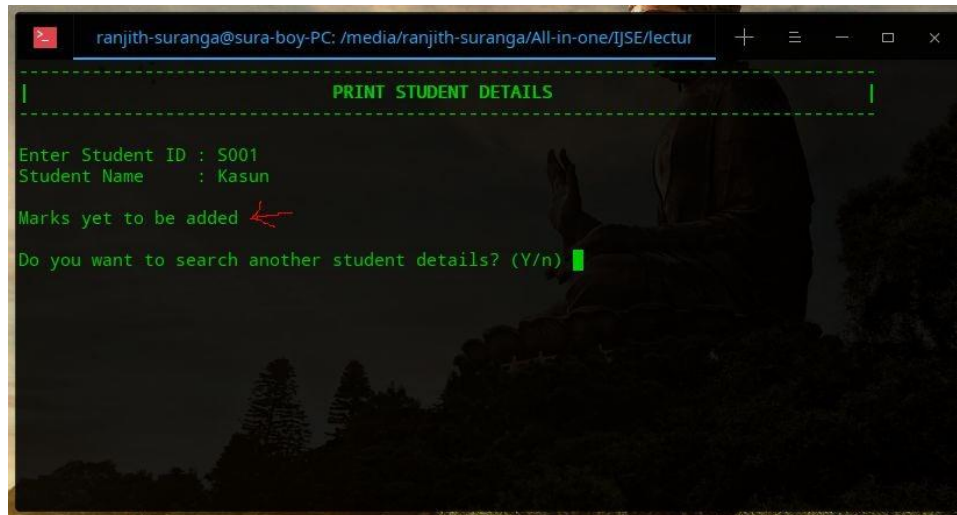


Figure 11 – Delete Student



## 7. Print Student Details

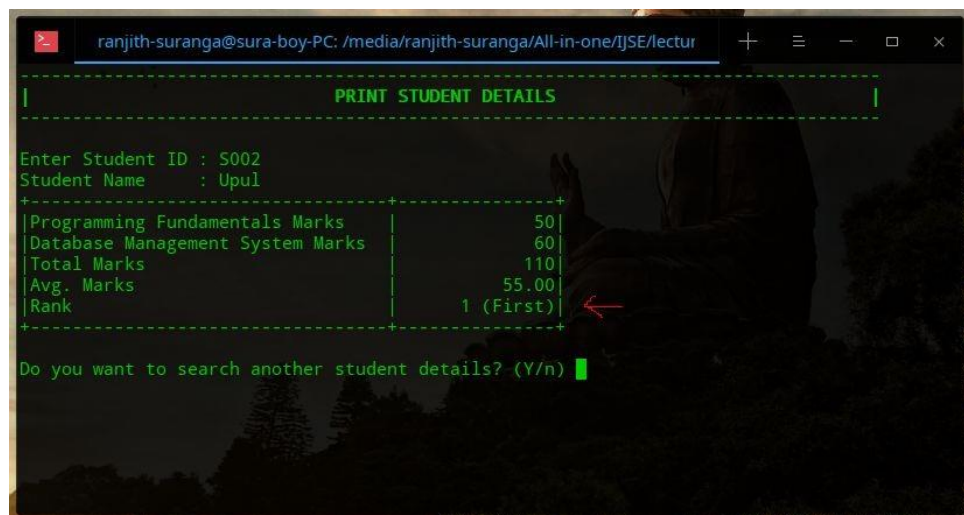
With this option, the user can view student details. First, he needs to enter a valid student id; otherwise, it should handle like previously. If the student marks haven't been added yet, then they should be displayed like below.



```
ranjith-suranga@sura-boy-PC: /media/ranjith-suranga/All-in-one/IJSE/lectur
|----- PRINT STUDENT DETAILS -----|
Enter Student ID : S001
Student Name      : Kasun
Marks yet to be added
Do you want to search another student details? (Y/n)
```

Figure 12 – Print Student Details (with validation)

If the student marks have been already added, then a comprehensive detail table should be displayed, which contains the marks, total marks, avg. marks as well as the rank of the student. First, Second, Third, and Last rank positions are also displayed in text close to the rank number in parenthesis. Once the information has been displayed, the user should be prompted whether to continue seeking student details or go back to the main menu.



```
ranjith-suranga@sura-boy-PC: /media/ranjith-suranga/All-in-one/IJSE/lectur
|----- PRINT STUDENT DETAILS -----|
Enter Student ID : S002
Student Name      : Upul
+-----+-----+
|Programming Fundamentals Marks|      50|
|Database Management System Marks|    60|
|Total Marks|                110|
|Avg. Marks|                55.00|
|Rank|                1 (First)|
+-----+-----+
Do you want to search another student details? (Y/n)
```

Figure 13 – Print Student Details (successful)

**Note:** More information about rank can be found in the next step.

## 8. Print Student Ranks

This displays all the students with their ranks. The students whose marks have not been added to the system yet doesn't display here. Apart from the ranks, their student Id, name, total marks, and avg. marks are also displayed here. It should prompt whether the user wants to stay in here or go back to the main menu.

```
ranjith-suranga@suraboy-PC: /media/ranjith-suranga/All-in-one/IJSE/lectures  
-----  
| PRINT STUDENTS' RANKS |  
-----  
+-----+-----+-----+-----+-----+  
| Rank | ID   | Name   | Total Marks | Avg. Marks |  
+-----+-----+-----+-----+-----+  
| 1     | S001 | Kasun  | 198         | 99.00      |  
| 2     | S005 | Ruwanthi | 180        | 90.00      |  
| 3     | S004 | Sachini | 140         | 70.00      |  
| 4     | S003 | Yasendra | 130        | 65.00      |  
| 5     | S002 | Nuwan  | 100         | 50.00      |  
+-----+-----+-----+-----+-----+  
Do you want to go back to main menu? (Y/n):
```

Figure 14 – Print Student Ranks

## 9. Best in Programming Fundamentals

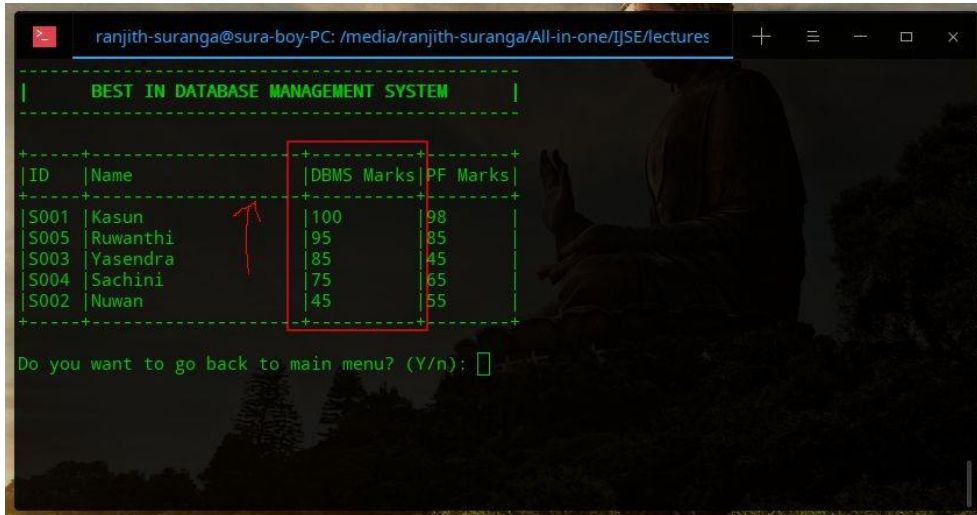
This displays who has done well in the Programming Fundamental module. Just like the previous one, only the students whose marks have been added take into the calculation. Just like above, in the end, it should prompt whether the user wants to stay in here or go back to the main menu.

```
ranjith-suranga@suraboy-PC: /media/ranjith-suranga/All-in-one/IJSE/lectures  
-----  
| BEST IN PROGRAMMING FUNDAMENTALS |  
-----  
+-----+-----+-----+-----+  
| ID   | Name   | PF Marks | DBMS Marks |  
+-----+-----+-----+-----+  
| S001 | Kasun  | 98       | 100        |  
| S005 | Ruwanthi | 85      | 95         |  
| S004 | Sachini | 65       | 75         |  
| S002 | Nuwan  | 55       | 45         |  
| S003 | Yasendra | 45      | 85         |  
+-----+-----+-----+-----+  
Do you want to go back to main menu? (Y/n):
```

Figure 15 – Best in Programming Fundamentals

## 10. Best in Database Management System [\[Demo\]](#)

This displays who has done well in the Database Management System module. Just like the previous one, only the students whose marks have been added take into the calculation. Just like above, in the end, it should prompt whether the user wants to stay in here or go back to the main menu.



```
ranjith-suranga@sura-boy-PC: /media/ranjith-suranga/All-in-one/IJSE/lectures
```

ID	Name	DBMS Marks	PF Marks
S001	Kasun	100	98
S005	Ruwanthi	95	85
S003	Yasendra	85	45
S004	Sachini	75	65
S002	Nuwan	45	55

Do you want to go back to main menu? (Y/n): ☐

Figure 16 – Best in DBMS

## Guidelines

- You may use any number of arrays. You may use matrix arrays if you require them. (This can be done without using matrix arrays)
- Expect for arrays you can't use any other Java data structure classes in Java SE framework like ArrayList, LinkedList, etc.
- You can't use any other 3rd party libraries or frameworks.
- You can't create classes except for the class that holds the main method.
- Use the **Scanner** class to get input from the command-line-interface.
- All validations that have been mentioned in this document should be implemented.
- It is not required to clear the command line screen while navigating between the options. But doing so highly recommend. [\[Demo\]](#)
- The code to clear the command line from inside a Java application is as follows. You can use this code when you need to clear the command line.

```
public final static void clearConsole() {  
    try {  
        final String os = System.getProperty("os.name");  
        if (os.contains("Windows")) {  
            new ProcessBuilder("cmd", "/c", "cls").inheritIO().start().waitFor();  
        } else {  
            System.out.print("\033[H\033[2J");  
            System.out.flush();  
        }  
    } catch (final Exception e) {  
        e.printStackTrace();  
        // Handle any exceptions.  
    }  
}
```

- You should write all the sorting algorithms by yourself. You should be able to explain the code that you have written.
- You can create as many methods as you wish in the only class that you have.
- Demo videos may help you to clarify your doubts to some extent.
- If you still have any questions, feel free to question.

### Evaluation Criteria

Method	Marks
Application	50
Area of knowledge (VIVA-VOCE)	50

**Pass Marks: 80**