

Case Study Demonstration

Data Structures using 'C'.

In growing competitive world, Computer programming is considered a basic and important skill for any student pursuing a career in engineering & technology

The objective of the course is to expose students with computational thinking and fill the gap between computer science fundamentals and programming skills.

The Data Structures with C course is designed from view of what is computation and how do you perform it, how do you write efficient code to make any system as automated and solve real-world problems. The principles of selection of topics and teaching methods are the basics of computer programming that should be introduced with enough details to practice computing with computers. The

course is developed with following objectives:

- 1. Storing and organizing data in a computer so that it can be used efficiently:** Students could have used computer for storing and organizing data manually but could have never done the same with programs, here the main goal is to expose the students about storing and organizing data through computer programs.
- 2. Implement programs that are efficient and fast to execute and also structure the data:** Computational thinking is a new concept need to be clearly introduced, including the basic concepts, what it is and is not. Given the problem statement students should solve it by writing the program but the written program has to be efficient and fast to execute.
- 3. Basic operations on data Structures like stacks, queue, linked list, trees and graphs:** Since this is a data structures lab, students need to understand the fundamental data structures and able to explain them without any difficulties.
- 4. Implementation of operations on stacks, queues, linked list, trees and graphs:** Understanding the basic concept theoretically is not a challenging task, but the challenge is to apply and solve the real-world problems with respect to their requirements.
- 5. Selection of Data Structures for a given application:** The criterion for selecting data structure for student practice is that the concepts are not only easy to be mastered, but also powerful enough to practice core conceptions and skills of computational thinking. Finally the students should be able to choose the appropriate method and technique in solving the real-world problem.

Student Information System(using Linked List in C)

Aim of Studying a Case Study is Students Getting familiar with program logic: The foremost important thing in programming is to understand the code and its logic i.e., flow of data with respect to logic of program. To make students understand the concepts clearly and to give some visual effects we use Turbo C++/Gcc Online compiler IDE to explain the code, the online IDE tool has a provision to stop the execution of program for a given break point and execute step by step by showing how the data is flowing in the program. This technique will help the student to understand program logic and data flow in program very easily, it also helps in identifying the faults in the program. IDE also helps in understanding difficult concepts like recursion, stack, queues, linked list etc.

Student Information System can be used by education institutes to maintain the records of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming. All these problems are solved using this Case Study.

Student structure

First we create a structure to store student data.

```
struct Student{  
    int rollnumber;  
    char name[100];  
    char phone[100];  
    float percentage;  
    struct Student *next;  
}*head;
```

head is a global pointer of type Student which will point to the first node in the linked list. The advantage of declaring head global is we can directly use it in any function without passing it as a parameter.

Insert Function

After that, we create a function insert, that will add a new node to the linked list. The insert function accepts student details as an argument. It creates a new node with the student details passed to the function and then inserts the new node at the beginning of the linked list.

```
void insert(int rollnumber, char* name, char* phone, float percentage)
{

    struct Student * student = (struct Student *) malloc(sizeof(struct Student));
    student->rollnumber = rollnumber;
    strcpy(student->name, name);
    strcpy(student->phone, phone);
    student->percentage = percentage;
    student->next = NULL;

    if(head==NULL){
        // if head is NULL
        // set student as the new head
        head = student;
    }
    else{
        // if list is not empty
        // insert student in beginning of head
        student->next = head;
        head = student;
    }

}
```

Search Function

The search function searches the record based on the roll number. The search function accepts 1 parameter that is the roll number of the student we want to search. The function traverses all the nodes of the linked list to find the required record.

```
void search(int rollnumber)
{
    struct Student * temp = head;
    while(temp!=NULL){
        if(temp->rollnumber==rollnumber){
            printf("Roll Number: %d\n", temp->rollnumber);
            printf("Name: %s\n", temp->name);
            printf("Phone: %s\n", temp->phone);
            printf("Percentage: %0.4f\n", temp->percentage);
            return;
        }
        temp = temp->next;
    }
    printf("Student with roll number %d is not found !!!\n", rollnumber);
}
```

Update Function

The update function first searches for the node with the required roll number. If the node is found, the program asks the user to enter new updated values.

```
void update(int rollnumber)
{

    struct Student * temp = head;
    while(temp!=NULL){

        if(temp->rollnumber==rollnumber){
            printf("Record with roll number %d Found !!!\n", rollnumber);
```

```

        printf("Enter new name: ");
        scanf("%s", temp->name);
        printf("Enter new phone number: ");
        scanf("%s", temp->phone);
        printf("Enter new percentage: ");
        scanf("%f",&temp->percentage);
        printf("Updation Successful!!!\n");
        return;
    }
    temp = temp->next;

}

printf("Student with roll number %d is not found !!!\n", rollnumber);

}

```

Delete Function

Delete works similar to search. We search for the record by it's roll number. If the record is found, we delete it from the linked list.

```

void Delete(int rollnumber)
{

    struct Student * temp1 = head;
    struct Student * temp2 = head;
    while(temp1!=NULL){

        if(temp1->rollnumber==rollnumber){

            printf("Record with roll number %d Found !!!\n", rollnumber);

            if(temp1==temp2){
                // this condition will run if
                // the record that we need to delete is the first node
            }
        }
        temp1 = temp1->next;
    }
}

```

```

        // of the linked list
        head = head->next;
        free(temp1);
    }
    else{
        // temp1 is the node we need to delete
        // temp2 is the node previous to temp1
        temp2->next = temp1->next;
        free(temp1);
    }

    printf("Record Successfully Deleted !!!\n");
    return;

}

temp2 = temp1;
temp1 = temp1->next;

}

printf("Student with roll number %d is not found !!!\n", rollnumber);

}

```

Display Function

Display function traverse the linked list and print all the details of each node of the linked list.

```

void display()
{
    struct Student * temp = head;
    while(temp!=NULL){

        printf("Roll Number: %d\n", temp->rollnumber);
        printf("Name: %s\n", temp->name);
        printf("Phone: %s\n", temp->phone);
    }
}

```

```

        printf("Percentage: %0.4f\n\n", temp->percentage);
        temp = temp->next;

    }
}

```

Online C Code:

```

/*****
****

```

Online C Compiler.

Code, Compile, Run and Debug C program online.

Write your code in this editor and press "Run" button to compile and execute it.

```

****
****/

```

```

#include<stdlib.h>
#include<string.h>
#include<stdio.h>
struct Student
{
    int rollnumber;
    char name[100];
    char phone[100];
    float percentage;
    struct Student *next;

}* head;
void insert(int rollnumber, char* name, char* phone, float percentage)
{

    struct Student * student = (struct Student *) malloc(sizeof(struct Student));
    student->rollnumber = rollnumber;

```

```

strcpy(student->name, name);
strcpy(student->phone, phone);
student->percentage = percentage;
student->next = NULL;

if(head==NULL){
    // if head is NULL
    // set student as the new head
    head = student;
}
else{
    // if list is not empty
    // insert student in beginning of head
    student->next = head;
    head = student;
}

}

void search(int rollnumber)
{
    struct Student * temp = head;
    while(temp!=NULL){
        if(temp->rollnumber==rollnumber){
            printf("Roll Number: %d\n", temp->rollnumber);
            printf("Name: %s\n", temp->name);
            printf("Phone: %s\n", temp->phone);
            printf("Percentage: %0.4f\n", temp->percentage);
            return;
        }
        temp = temp->next;
    }
    printf("Student with roll number %d is not found !!!\n", rollnumber);
}

void update(int rollnumber)

```



```

{

struct Student * temp = head;
while(temp!=NULL){

    if(temp->rollnumber==rollnumber){
        printf("Record with roll number %d Found !!!\n", rollnumber);
        printf("Enter new name: ");
        scanf("%s", temp->name);
        printf("Enter new phone number: ");
        scanf("%s", temp->phone);
        printf("Enter new percentage: ");
        scanf("%f",&temp->percentage);
        printf("Updation Successful!!!\n");
        return;
    }
    temp = temp->next;

}

printf("Student with roll number %d is not found !!!\n", rollnumber);

}

void Delete(int rollnumber)
{
    struct Student * temp1 = head;
    struct Student * temp2 = head;
    while(temp1!=NULL){

        if(temp1->rollnumber==rollnumber){

            printf("Record with roll number %d Found !!!\n", rollnumber);

            if(temp1==temp2){
                // this condition will run if

```

```

        // the record that we need to delete is the first node
        // of the linked list
        head = head->next;
        free(temp1);
    }
    else{
        // temp1 is the node we need to delete
        // temp2 is the node previous to temp1
        temp2->next = temp1->next;
        free(temp1);
    }

    printf("Record Successfully Deleted !!!\n");
    return;

}
temp2 = temp1;
temp1 = temp1->next;

}
printf("Student with roll number %d is not found !!!\n", rollnumber);

}
void display()
{
    struct Student * temp = head;
    while(temp!=NULL){

        printf("Roll Number: %d\n", temp->rollnumber);
        printf("Name: %s\n", temp->name);
        printf("Phone: %s\n", temp->phone);
        printf("Percentage: %0.4f\n\n", temp->percentage);
        temp = temp->next;
    }
}

```

```

    }
}
int main()
{
    head = NULL;
    int choice;
    char name[100];
    char phone[100];
    int rollnumber;
    float percentage;
    printf("1 to insert student details\n2 to search for student details\n3 to delete student
details\n4 to update student details\n5 to display all student details");
    do
    {
        printf("\nEnter Choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                printf("Enter roll number: ");
                scanf("%d", &rollnumber);
                printf("Enter name: ");
                scanf("%s", name);
                printf("Enter phone number: ");
                scanf("%s", phone);
                printf("Enter percentage: ");
                scanf("%f", &percentage);
                insert(rollnumber, name, phone, percentage);
                break;
            case 2:
                printf("Enter roll number to search: ");
                scanf("%d", &rollnumber);
                search(rollnumber);
                break;

```

```

        case 3:
            printf("Enter roll number to delete: ");
            scanf("%d", &rollnumber);
            Delete(rollnumber);
            break;
        case 4:
            printf("Enter roll number to update: ");
            scanf("%d", &rollnumber);
            update(rollnumber);
            break;
        case 5:
            display();
            break;
    }

    } while (choice != 0);
}

```

Output:

1 to insert student details
 2 to search for student details
 3 to delete student details
 4 to update student details
 5 to display all student details

Enter Choice: 1

Enter roll number: 10

Enter name: Ravi

Enter phone number: 9012132XXX

Enter percentage: 98.8

Enter Choice: 1

Enter roll number: 14

Enter name: Rohan

Enter phone number: 908737XXXX

Enter percentage: 87.9

Enter Choice: 1

Enter roll number: 9

Enter name: Roshan

Enter phone number: 878920XXXX

Enter percentage: 84.3

Enter Choice: 1

Enter roll number: 5

Enter name: Ravil

Enter phone number: XXXX891298

Enter percentage: 54.8

Enter Choice: 5

Roll Number: 5

Name: Ravil

Phone: XXXX891298

Percentage: 54.8000

Roll Number: 9

Name: Roshan

Phone: 878920XXXX

Percentage: 84.3000

Roll Number: 14

Name: Rohan

Phone: 908737XXXX

Percentage: 87.9000

Roll Number: 10

Name: Ravi

Phone: 9012132XXX

Percentage: 98.8000

Enter Choice: 2

Enter roll number to search: 9

Roll Number: 9

Name: Roshan

Phone: 878920XXXX

Percentage: 84.3000

Enter Choice: 3

Enter roll number to delete: 9

Record with roll number 9 Found !!!

Record Successfully Deleted !!!

Enter Choice: 4

Enter roll number to update: 10

Record with roll number 10 Found !!!

Enter new name: Shiv

Enter new phone number: 7777XXXXXX

Enter new percentage: 86.8

Updation Successful!!!

Enter Choice: 5

Roll Number: 5

Name: Ravil

Phone: XXXX891298

Percentage: 54.8000

Roll Number: 14

Name: Rohan

Phone: 908737XXXX

Percentage: 87.9000

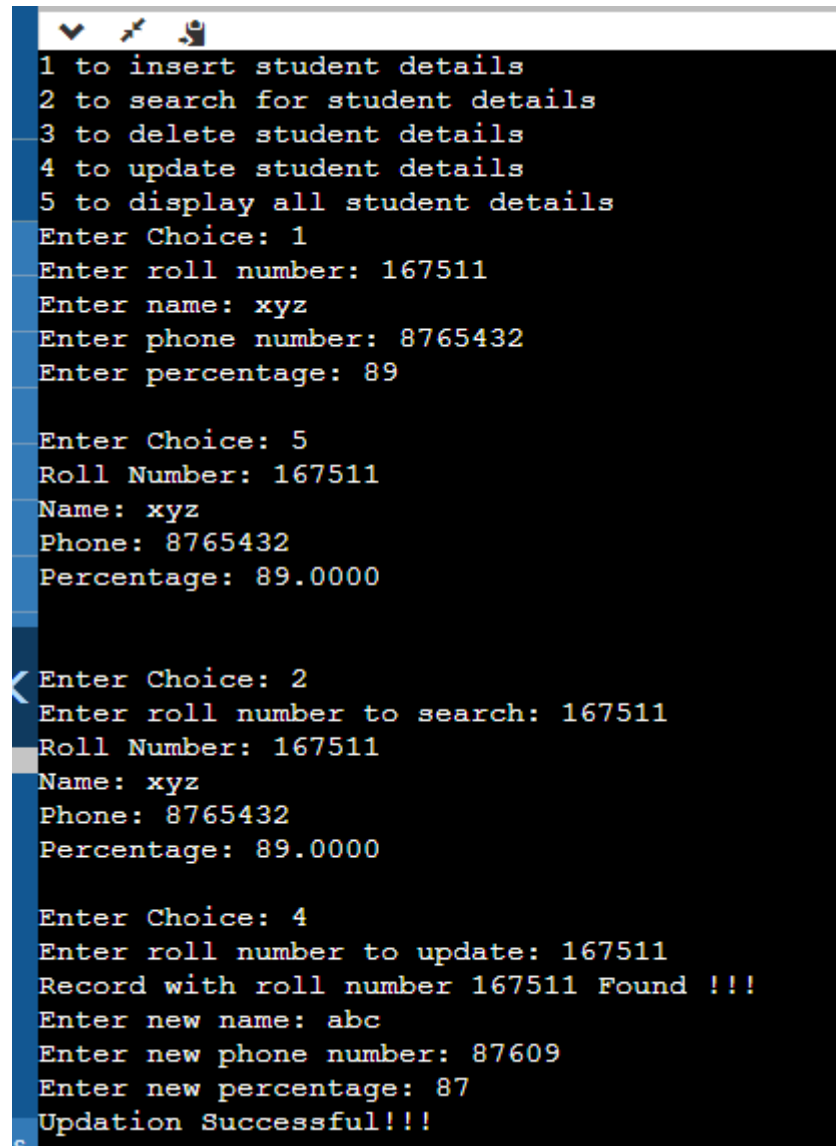
Roll Number: 10

Name: Shiv

Phone: 7777XXXXXX

Percentage: 86.8000

Enter Choice: 0



```
1 to insert student details
2 to search for student details
3 to delete student details
4 to update student details
5 to display all student details
Enter Choice: 1
Enter roll number: 167511
Enter name: xyz
Enter phone number: 8765432
Enter percentage: 89

Enter Choice: 5
Roll Number: 167511
Name: xyz
Phone: 8765432
Percentage: 89.0000

Enter Choice: 2
Enter roll number to search: 167511
Roll Number: 167511
Name: xyz
Phone: 8765432
Percentage: 89.0000

Enter Choice: 4
Enter roll number to update: 167511
Record with roll number 167511 Found !!!
Enter new name: abc
Enter new phone number: 87609
Enter new percentage: 87
Updation Successful!!!
```

B.The works must be available for peer review and critique (4)

A meeting was conducted with the Departmental Advisory committee (DAC) and staff members for the awareness purpose and as well as for review and critique.

S.No	Name of the Faculty	Designation	Suggestions/Review/Remark	Signature

C. The work must be reproducible and developed further by other scholars (2)

This case study demonstration was helpful for students to gain insight into the subject and it was further carried out **by Ms.Ishrath Nousheen to teach OOPs through C++ course.**

D. Statement of clear goals, use of appropriate methods, significance of results, effective presentation and reflective critique (10)**Goals:**

A Goal of teaching with case studies is that the students are actively engaged in figuring out the principles by abstracting from the examples. This develops their skills in:

- Problem solving
- Analytical tools, quantitative and/or qualitative, depending on the case
- Decision making in complex situations
- Coping with ambiguities

Appropriate Methods: ICT, PPTs Presentation (Seminars), Online c compiler, Turbo C++.

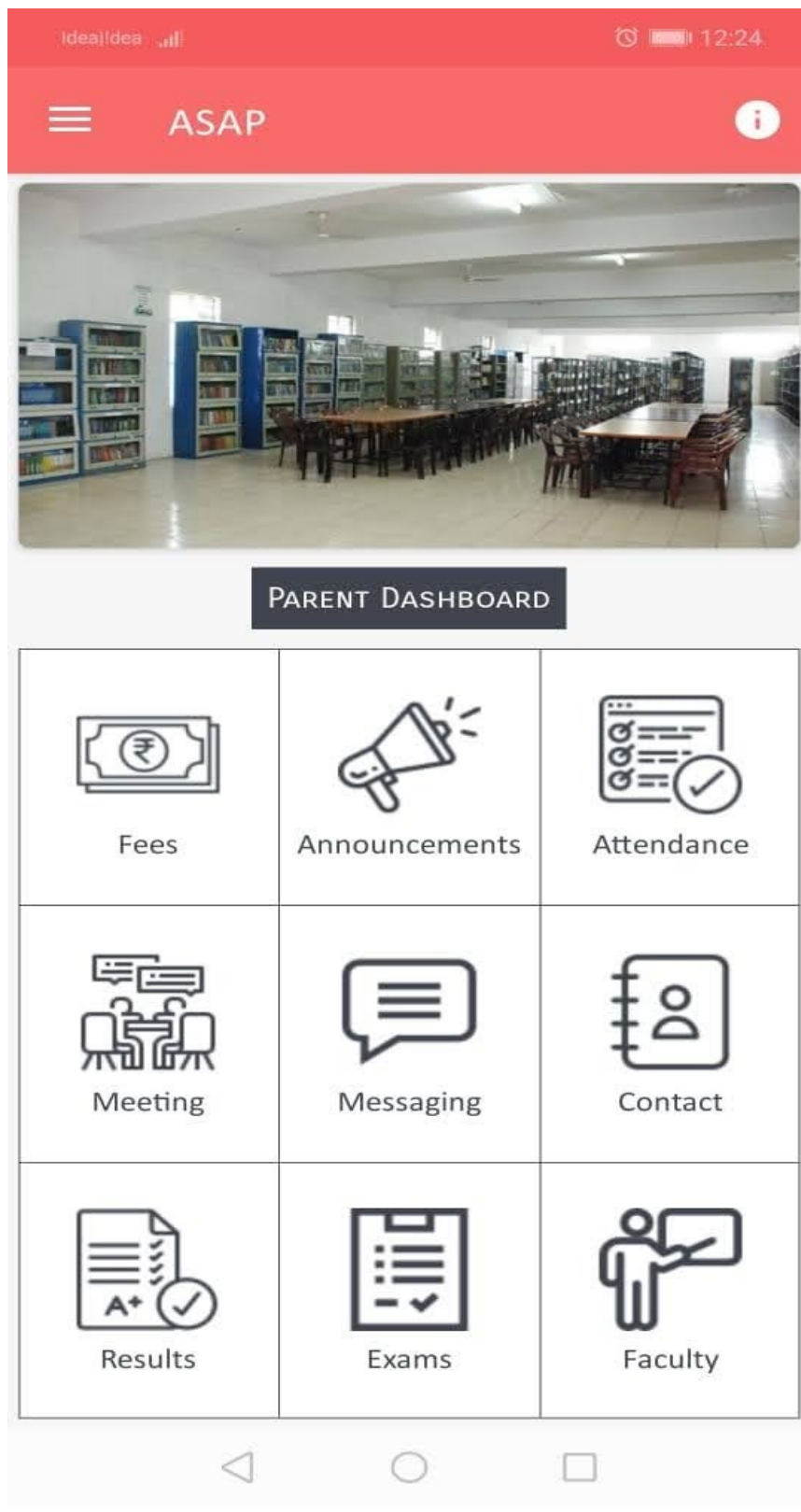
Significance of Result:

The main objective in mind when teaching this course is to make programming more interesting to students who do not view it so, assuming that greater student interest would lead to better performance in class and deeper understanding and appreciation of the subject. Introductory programming courses often teach such basic concepts that it is difficult to design assignments that are simultaneously simple, challenging and interesting. We feel that our teaching methodology provides all three aspects. Our interactions in the classroom, along with the average performance of the students in the assignments, provide encouraging evidence that our approach worked well in these aspects.

Students were able to get good results both in practical and theory

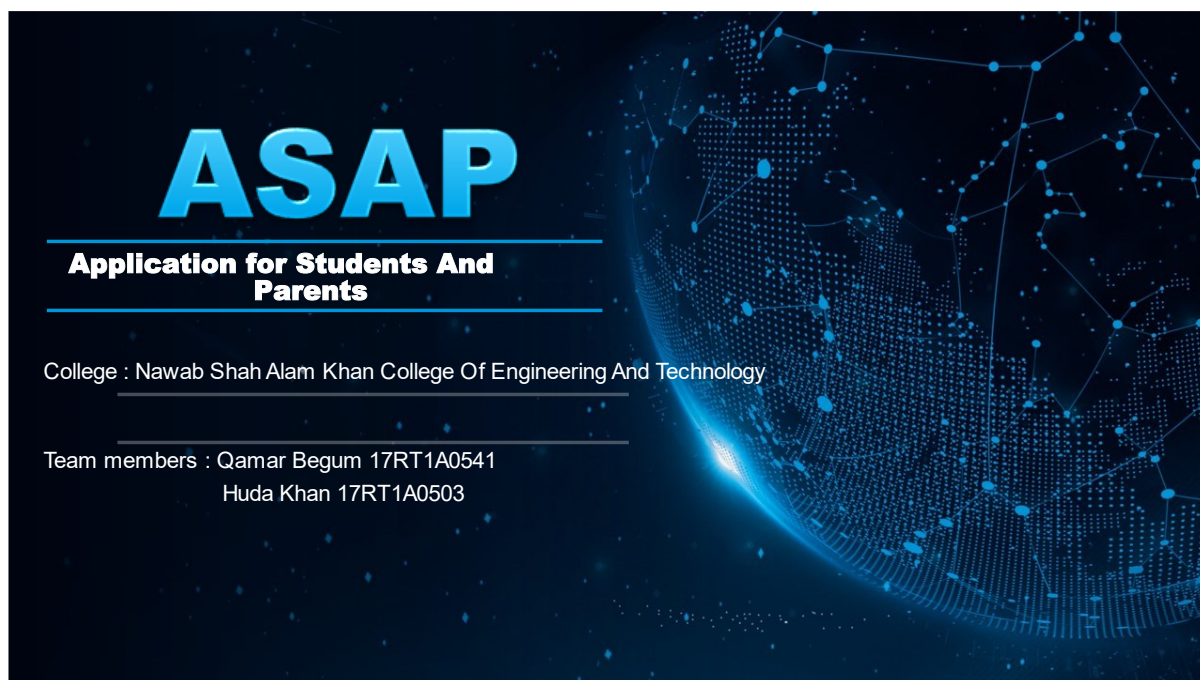
	Practical Exam	Theory semester end Exam
Pass percentage	100%	86%

Students are able to develop a Software Application entitle”**ASAP-Application for Students and Parents**”.



Effective presentation and Reflective critique:

The objective of the course is providing a foundation of programming principles and data structures that student can and will apply to discipline study. Our goal is to involve student actively in using programming in their discipline and seek a unified approach to teach various non-majors. To evaluate the effectiveness of our approach, during the course we interacted with student personally and took the response about various aspects regarding the course.



Feedback and Result Analysis

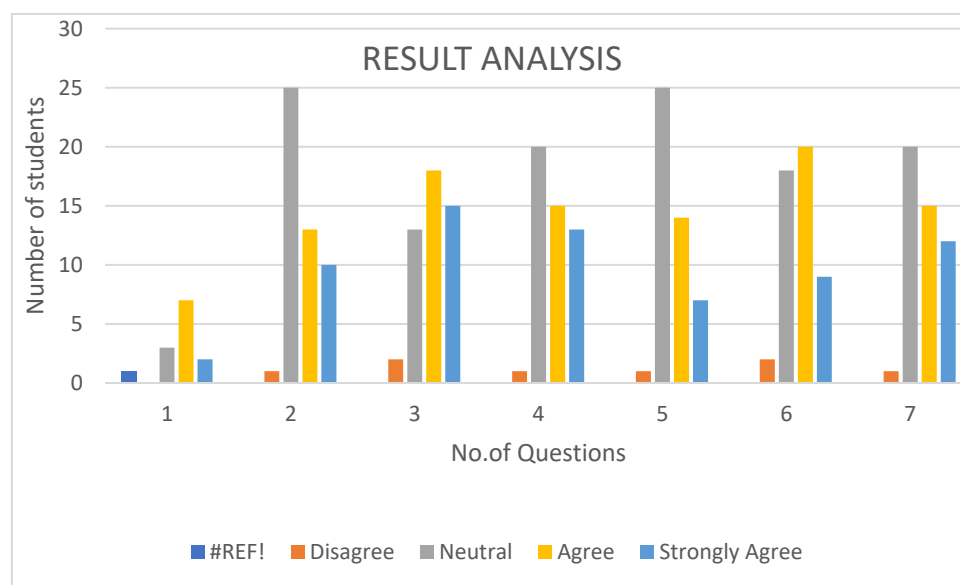
A survey has been conducted in which students and faculties were asked certain questions .

Questionnaire/Feedback From Students

S.No	Question No	Questions
1	Q1	The Case Study increased your knowledge and skills in the subject matter.
	Q2	The course gave you the confidence to do more advanced work in the subject
	Q3	Do you believe that what you are being asked to learn in this course is important
	Q4	Overall, this course met your expectations for the quality of the course
	Q5	The course was helpful in progress toward my degree
	Q6	The instructor's teaching methods were effective.
	Q7	The instructor encouraged student participation in class.

Distribution of students' answers (numbers and percentage) provided for the survey's questions

Question No	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Q1	1	2	20	15	12
Q2	1	1	25	13	10
Q3	2	2	13	18	15
Q4	1	1	20	15	13
Q5	2	1	25	14	7
Q6	1	2	18	20	9
Q7	1	1	20	15	12



Questionnaire/Feedback from Faculty/Peer Review

S.No	Question No	Questions
1	Q1	The instructor effectively explained and illustrated the purpose and importance of the course concepts
	Q2	The instructor communicated clearly and was easy to understand i.e., teaching methods were effective.
	Q3	The instructor effectively organized and facilitated well-run learning activities and was well-prepared for class/discussion sections.
	Q4	Did this Innovative practice helped to improve and increase the abilities of students.
	Q5	The instructor cared about the students, their progress, and successful course completion.
	Q6	I would highly recommend this instructor to other students as it encouraged student participation in class.

Question No	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Q1	0	0	5	2	5
Q2	0	0	2	6	4
Q3	0	0	3	6	3
Q4	0	0	2	5	5
Q5	0	0	1	5	6
Q6	0	0	1	3	8

