K. J. Somaiya College of Engineering, Mumbai-77

(A Constituent college of Somaiya Vidyavihar University)

Batch: A2 Roll No.:16010123032

Experiment No. 1

Grade: AA / AB / BB / BC / CC / CD /DD

Title: Implementation of Abstract Data Type

Objective: Implementation of ADT without using any standard library function

Expected Outcome of Experiment:

CO	Outcome
CO 1	Explain the different data structures used in problem solving.

Books/ Journals/ Websites referred:

www.google.com
https://www.programiz.com/c-programming/online-compiler/
https://www.geeksforgeeks.org/printf-in-c/
https://www.w3schools.com/

Abstract:-

(Define ADT. Why are they important in data structures?)

1)Encapsulation

2)Abstraction

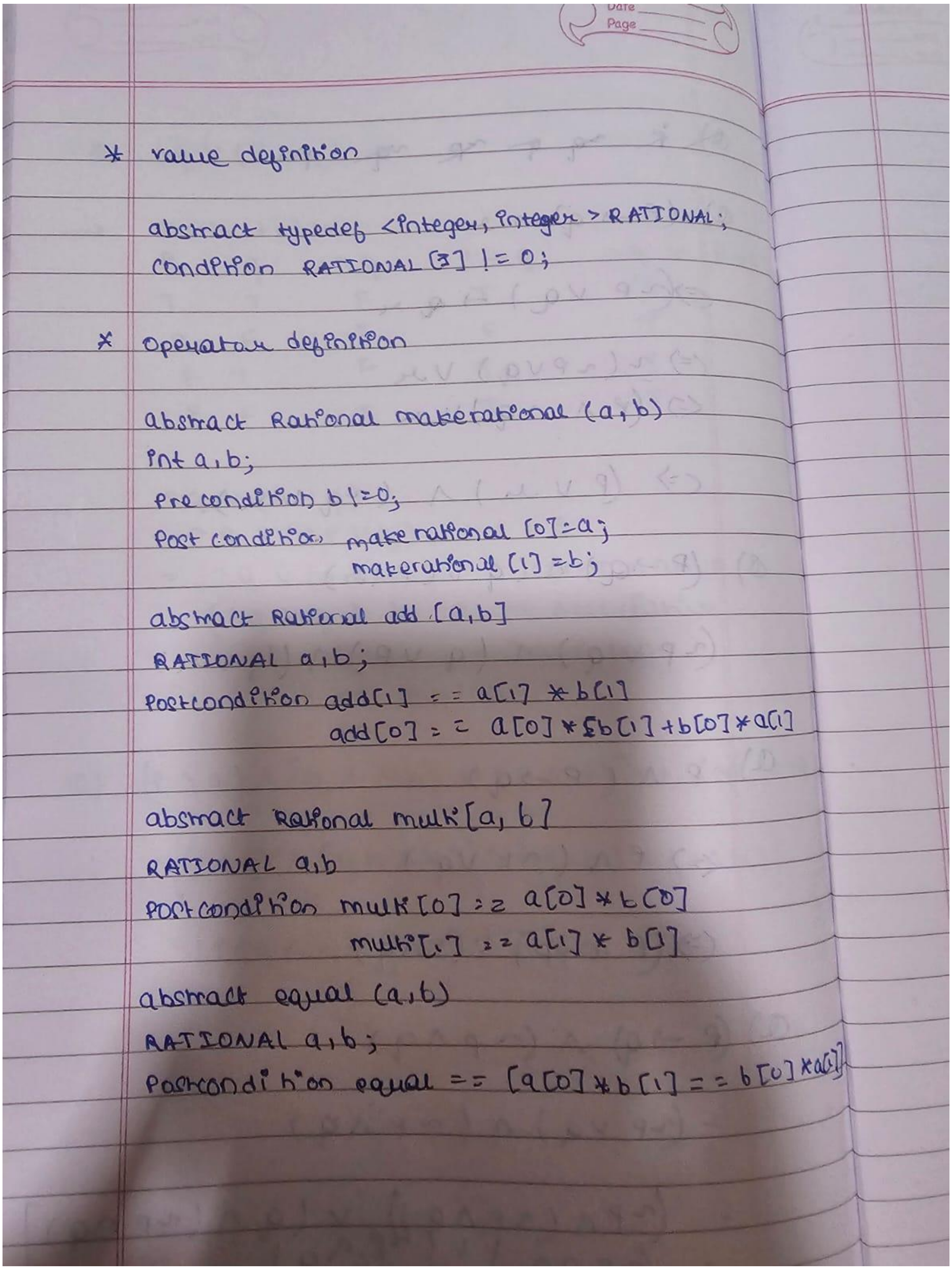
3)Data Structure Independece

4)Information Hinding

5)Modularity

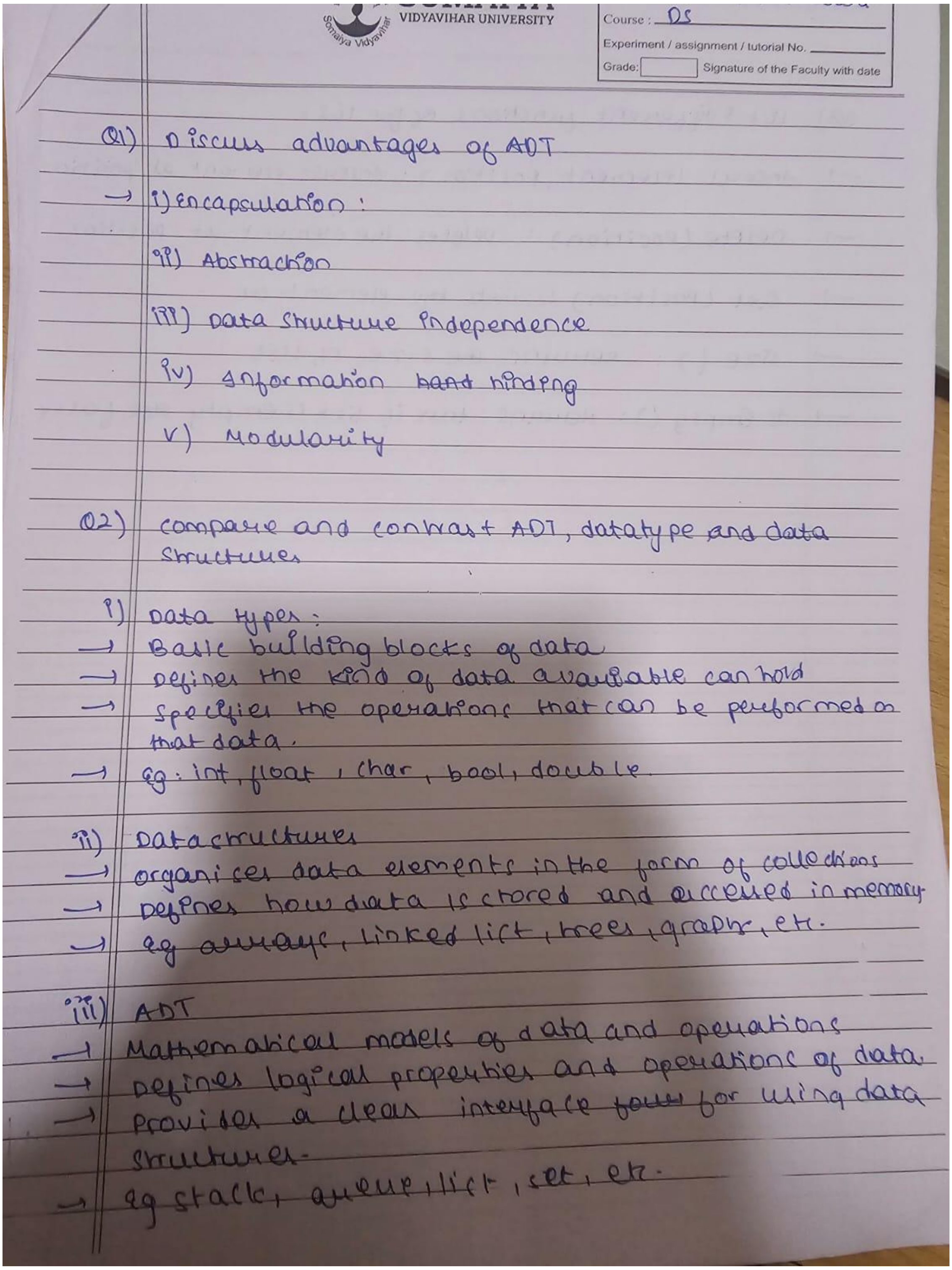
Abstract Data Type for <u>rational</u>

<u>(Value and Operator definition)</u>



```
* value definition

  abstract typedef <integer, integer > RATIONAL;
  condition RATIONAL [1] != 0;

* operator definition

  abstract Rational makerational (a, b)
  int a, b;
  precondition b!=0;
  post condition makerational [0]=a;
                 makerational [1] =b;

  abstract Rational add [a,b]
  RATIONAL a,b;
  postcondition add[1] == a[1] * b[1]
                add[0] == a[0] * £b[1] + b[0] * a[1]

  abstract Rational multi[a, b]
  RATIONAL a,b
  postcondition multi[0] == a[0] * b[0]
                multi[1] == a[1] * b[1]

  abstract equal (a,b)
  RATIONAL a,b;
  postcondition equal == [a[0] * b[1] == b[0] * a[1]]
```

Post lab questions:



VIDYAVIHAR UNIVERSITY

Course : DS

Experiment / assignment / tutorial No. ______

Grade: □ Signature of the Faculty with date

Q1) Discuss advantages of ADT

→ i) Encapsulation :

ii) Abstraction

iii) Data structure independence

iv) Information ~~hand~~ hinding

v) Modularity

Q2) compare and contrast ADT, datatype and data structures

i) Data types :

→ Basic building blocks of data

→ Defines the kind of data available can hold

→ Specifies the operations that can be performed on that data.

→ eg: int, float, char, bool, double.

ii) Data structures

→ organises data elements in the form of collections

→ Defines how data is stored and accessed in memory

→ eg arrays, linked list, trees, graphs, etc.

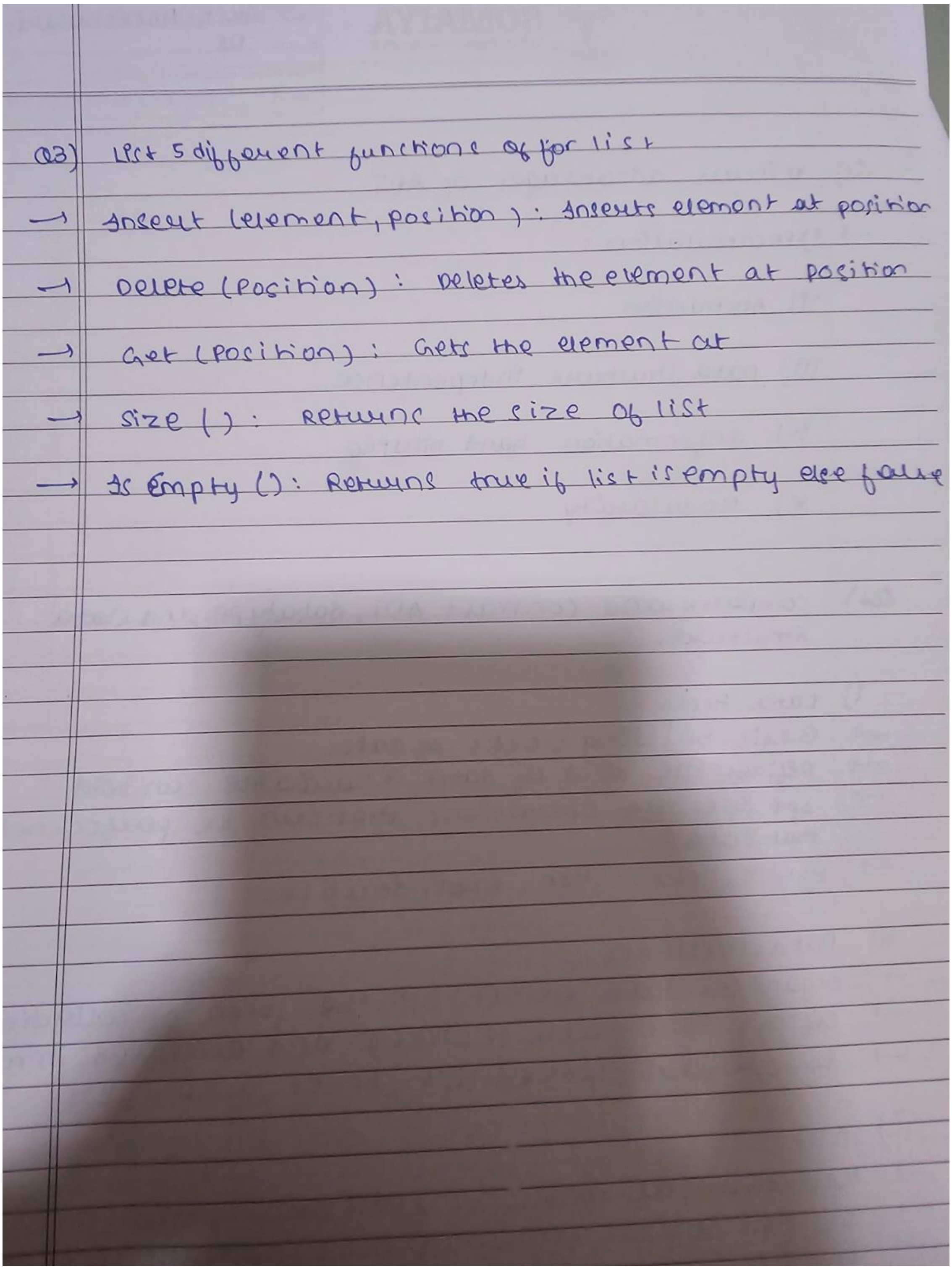
iii) ADT

→ Mathematical models of data and operations

→ Defines logical properties and operations of data

→ Provides a clear interface ~~fous~~ for using data structures.

→ eg stack, queue, list, set, etc.



Q3) List 5 different functions of for list

→ Insert (element, position) : Inserts element at position

→ Delete (position) : Deletes the element at position

→ Get (position) : Gets the element at

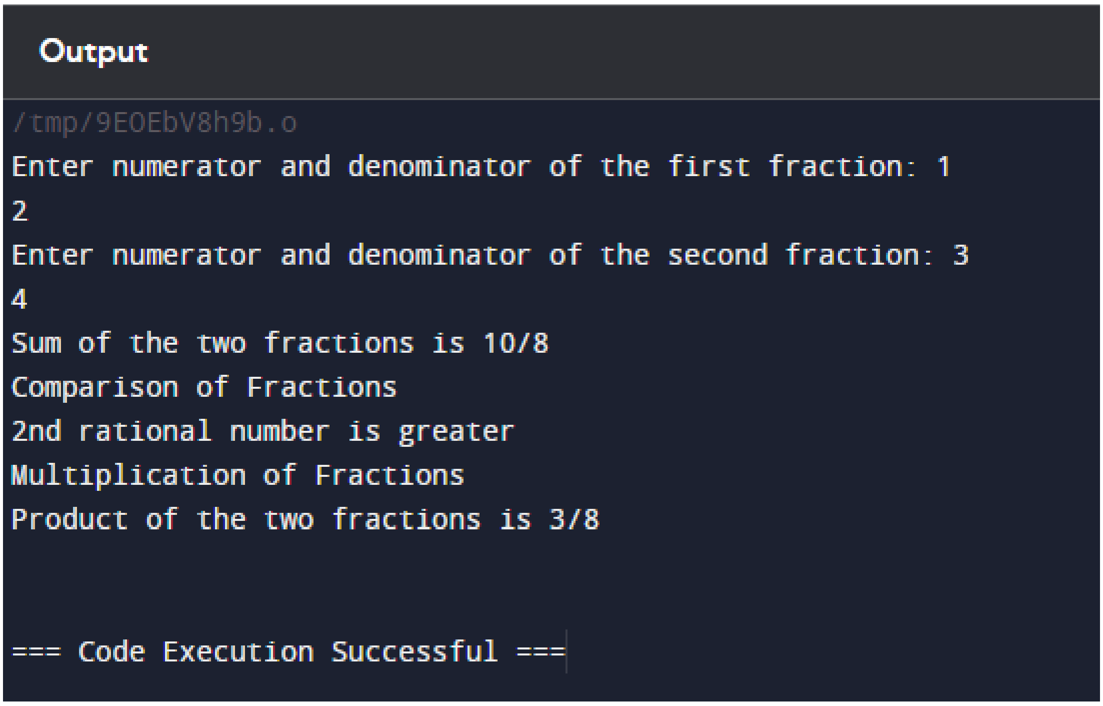
→ size ( ) : Returns the size of list

→ Is Empty (): Returns true if list is empty else false

Program code and Output screenshots:

```
// Online C compiler to run C program online
#include <stdio.h>
#include <stdlib.h>
#include <math.h>


  typedef struct{
    int den;
    int num;

  }rational;

  rational comparison(rational r1, rational r2){
    if ((r1.num*r2.den)>(r1.den*r2.num)){
      printf("\n1st rational number is greater\n");
    }
    if ((r1.num*r2.den)<(r1.den*r2.num)){
      printf("\n2nd rational number is greater\n");
    }
    if ((r1.num*r2.den)==(r1.den*r2.num)){
      printf("\nboth are equal\n");
    }
  }
```

```
rational multiply(rational r1, rational r2) {
rational result;
result.num = r1.num * r2.num;
result.den = r1.den * r2.den;

return result;
}

rational add(rational r1, rational r2){
    rational result;
    if (r1.den!=r2.den){
        result.num=(r1.num*r2.den)+(r1.den*r2.num);
        result.den=(r1.den*r2.den);
    }
    else{
        result.num=r1.num+r2.num;
        result.den=r1.den;
    }
printf("Sum of the two fractions is %d/%d\n", result.num, result.den);
}
int main() {
rational first, second;

printf("Enter numerator and denominator of the first fraction: ");
scanf("%d %d", &first.num, &first.den);
printf("Enter numerator and denominator of the second fraction: ");

scanf("%d %d", &second.num, &second.den);

rational sum = add(first, second);

printf("Comparison of Fractions");

comparison(first, second);

printf("Multiplication of Fractions\n");

rational product = multiply(first, second);

printf("Product of the two fractions is %d/%d\n", product.num, product.den);

return 0;

}
```



Output

```
/tmp/9EOEbV8h9b.o
Enter numerator and denominator of the first fraction: 1
2
Enter numerator and denominator of the second fraction: 3
4
Sum of the two fractions is 10/8
Comparison of Fractions
2nd rational number is greater
Multiplication of Fractions
Product of the two fractions is 3/8


=== Code Execution Successful ===
```

Conclusion:-

Learning about Abstract Data Types (ADTs), especially list ADTs, has provided a solid grounding in the principles of data structures. Understanding the abstract operations related to lists has enhanced my ability to effectively visualize and manage data. This fundamental knowledge is crucial as it sets the stage for understanding more complex data structures and algorithms.