Name - Gaurav pachauri
Roll No - 2404400062 (15)
Section - C

1. Handling Division by Zero :- subject - C# lab.

Task - Read two numbers and divide them.
if the denominator is zero → catch error and show a message.
finally block → runs always, even if there's an error.

ex -
```
using System;

class Program {
    static void Main () {
        try {
            console.Write ("enter numerator: ");
            int num = Convert.ToInt32(Console.ReadLine());
            Console.Write ("Enter denominator: ");
            int den = Convert.ToInt32(Console.Readline());
            int result = num|den;
            Console.WritiLine ("Result: " + result );
        }
        catch ( DivideByZeroExpection ) {
            Console.Writeline ("Division by zero is not Allowed");
        }
        finally {
            Console.WriteLine ("Execution completed");
        }
    }
}
```

2. Multiple Catch Blocks :-

Handle 3. possible error:

- FormatException → Wrong input type
- OverflowException → input too big.
- General Exception → any other issue.

ex.

```
try {
      Console.Write("Enter a number:");
      int num = Convert.ToInt32(Console.ReadLine());
      Console.WriteLine("You entered: "+num);
  }
   Catch (FormatException) {
       Console.WriteLine("Invalid format! Enter only no.");
  }
  Catch (OverflowException) {
       Console.WriteLine("Number too Large!");
   }
   Catch (Exception) {
        Console.WriteLine("Some other error occured");
   }
```

3. Custom Exception - NegativeSalaryException -

if salary<0, throw your own exception.

```
using System;
  Class NegativeSalaryException : Exception {
          public NegativeSalaryException (string message):
           base(message) { }
     }
```

```
Class Porogram {
    Static void Main () {
        Console.Write ("Entor Salary: ");
        double salary = Convert.ToDouble (Console.ReadLine());
    try {
        if (salary<0)
            throw new NegativeSalaryException ("Salary cannot
            be negative");
          console.WriteLine ("Salary is valid:"+ Salary);
      }
      Catch (NegativeSalaryException e) {
            Console.WriteLine (e.message);
        }
      }
    }
```

Q-4. Banking Scenorio - InsufficientBalanceException -

if withdrawol > balance → throw error.

ex-

```
using System;
Class InsufficientBalanceException: Exception {
    public InsufficientBalanceException (String message):
    base (message) { }
}
Class Bank {
    Static void Main () {
     double balance = 5000;
Consale.Worite ("entor Withdrawl amount :");
double amount= convert.ToDouble Console.Readline());
try{
    if (amount > balance)
        throw new InsufficientBalanceException ("Insufficeint
            balance");
    balance -= amount ;
    Consale.Writeline ("Withdrawl succeseful. Remaining
        balance;" + balance);
   }
   catch (InsufficeintBalanceException e){
        Consale.Wouteline (e.message);
   }
  }
}
```

5. Student Marks validation :-

Marks should be 0-100. if not throw exception.

ex -

```
using System;
class InvalidMarksException : Exception {
   public InvalidMarksException (String message): base(message){}
}
  class Student {
      public int Marks{
          get; set;
      }
  public void Setmearks(int marks){
   if (marks<0 || marks >100)
throw new InvalidMarksException("Marks must between
        0 and 100!");
    Marks = marks;
  }
  }

 Class Porogram {
    Static void Main (){
      Student s = new Student ();
     try {
        Console.Write ("enter marks: ");
      int m = Convert.ToInt32 ( Console.Readline ());
        S. setmarks(m);
      Console.Writeline ("valid marks entered!"+ Marks);
    }
   Catch (InvalidMarksException e){
          Console.Writeline (e.message);
      }
    }
  }
```

MCQ Answers :-

1. B - try handles exception
2. C - finally always Run
3. B - Exception is the base class
4. A - program crashes abnormally
5. B - throw is used to raise Exception
6. C - divide by zero - DivideByZeroException
7. B - Specific catch before general
8. B - finally can exist without Catch
9. B - prints 'Division by zero Not Allowed
10. A - Accessing outside Array - IndexOutofRangeException.
11. A - rethrows same exception.
12. B - " Index error" then End of program
13. B - used for user defined exceptions
14. B - Invalid number formate.
15. C - catch runs on error
16. True - custom exception inherit from exception
17 B - passing up the call stack
18. D - catch and finally are optional
19. B - Finaly return overide try's.
20 A - must inherit from exception or Application Exception