IARE

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING
(An Autonomous Institute affiliated to JNTUH, Hyderabad)
Dundigal, Hyderabad - 500 043

LABORATORY WORK BOOK

Name of the Student : Mahek Jabeen

Class : CSE – C Semester : III

Course Code : AITD02 Course Name : PWO Lab

Roll Number									
2	3	9	5	1	A	0	5	4	U

Name of the Course Faculty : Mr. P. Suresh Kumar Faculty ID : IARE 11068

Exercise Number : 3 Week Number : 3 Date : 29/11/24

S. No.	Exercise Number	EXERCISE NAME	MARKS AWARDED					
			Aim/ Preparation	Algorithm / Procedure / Performance in the Lab	Source Code / Calculations and Graphs	Program Execution / Results and Error Analysis	Viva - Voce	Total
			4	4	4	4	4	20
1	3.1	Find the Bomb	4	4	4	4	3	19
2	3.2	finding Nemo						
3	3.3	A week After						
4								
5	3.4	Track the Robot						
6	3.5	True Alphabetic Order						
7	3.6	Largest Common Ending						
8	3.7	Clear Brackets						
9	3.8	Count the number of duplicate characters						
10	3.9	Uban Numbers.						
11								
12								

Mahek
Signature of the Student

P. Suresh
Signature of the Faculty

START WRITING FROM HERE

3.1 Find the Bomb

```
using System;
public class program
{
    Public static void main()
    {
        Console.WriteLine(Bomb("There is a bomb"));
        Console.WriteLine(Bomb("Hey, did you think there is
                                a bomb?"));
        Console.WriteLine(Bomb("This goes boom!!!"));
        Console.ReadKey();
    }
    Public static string Bomb(String txt)
    {
        String lowerCaseTxt = txt.ToLower();
        if(lowerCase.Txt.Contains("bomb"))
        {
            return "DUCK!!!";
        } else {
            return "There is no bomb, relax.";
        }
    }
}
```

Output

DUCK!!!
DUCK!!!
There is no bomb, relax.

ROLL NUMBER :

3.2. Finding Nemo

```
using system;
class Program
{
    Public static string findNemo(string Sentence)
    {
      String [] arr = Sentence.split(' ');
      return (arr.contains("Nemo")); "I found Nemo at
      {array.IndexOf(arr."Nemo") + 1}!":"I can't find
                                        Nemo:(";
    }
     Public static void main(){
          Console.WriteLine(Find Nemo("I am finding Nemo!"));
          Console.WriteLine(Find Nemo("Nemo is me"));
          Console.WriteLine(find nemo("I am Nemo"));
          Console.WriteLine(Find Nemo("I am looking for Nemo"));
          Console.WriteLine(find Nemo("There's no such word"));
          Console.ReadKey();
     }
}
```

Actual Output:

I found Nemo at 4!
I found Nemo at 1!
I found Nemo at 2!
I found Nemo at 5!
I can't find Nemo:(.

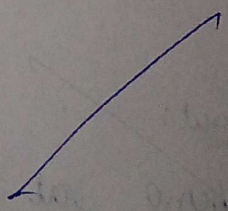
ROLL NUMBER :

3.3 A week after

```
using System;
using System.Globalization;
Public class Program {
    public static void main() {
        Console.WriteLine(Week After ("12/03/2020"));
        Console.WriteLine(Week After ("21/12/1989"));
        Console.WriteLine(Week After ("22/07/2024"));
        Console.ReadKey();
    }
    Public static string weekAfter(String date) {
        string format = "dd/mm/yyyy";
    DateTime passeddate = DateTime.parseExact(date,
            format, Culture Info.Invarient(culture);
    datetime weeklater = Passed date.AddDays(z);
    return weekhater.To string(format, culture
            Info.Invarient culture);
    }
}
```

Actual Output:

19/03/2020
28/12/1989
29/07/2024.



3.A Track the Robot

```
using system;
class program
{
    static void main()
    {
        int[] position1 = TrackRobot(new String[] {"right to",
                                "up 50", "left 30", "down 10"});
        Console.WriteLine($"final Position : {Position1[0]}
                                {Position1[1]}");
        int[] position2 = TrackRobot(new string[] {});
        Console.WriteLine($"Final Position : [{Position2[0]}, {Position
                                [1]}]");
        int[] position3 = TrackRobot(new string[] {"right 100",
                                "right 100", "up 500", "up 10,000"});
        Console.WriteLine($"Front position ({Position 3[0]}
                                {Position 3[1]}]");

        Console.ReadKey();
    }
    Static int[] TrackRobot(string[] instructions)
    {
        int x = 0;
        int y = 0;
        for Each (var instruction.Spirit(" ");
        string direction = Parts[0];
        int distance = Int.Parse(Parts[1]);
        Switch(direction)
        {
            Case 'right':
            xt = distance;
            break;
```

```
            case 'left':
                x = distance;
                break;
            case "up";
                yt = distance;
                break;
            case "down";
                y = distame;
                break;
            default;
            throw how new ArgumentException("Invalid direction");
        }
    }
    return new int[]{x, y};
  }
}
```

Output:

Final position:[-20,40]
Final Position:[0,0]
final Position:[200, 10500].

3.5 True Alphabetical Order.

```
using System;
using System.Linq;
Public class Program
{
    Public static void Main()
    {
        Console.WriteLine(TrueAlphabetical("hello world"));
        Console.WriteLine(TrueAlphabetical("care is awesome"));
        Console.WriteLine(TrueAlphabetical("have a nice day"));
        Console.ReadKey();
    }
    Public static string TrueAlphabetic(string input)
    {
```

```
        var sortedchars = new string input.Replace(" ","").
                                        OrderBy(c=>c.ToArray());
        var result = "";
        var index = 0;
        foreach(var word in input.split(" "));
        {
            if(result.length>0)
            {
                result += " ";
            }
            result += sortedchars.substring(index, word.Length);
            index += word.length;
        }
        return result;
    }
}
```

Output:

```
dehil looru
aabdee ei inosstw
aaac d eehi nvy.
```

3.6 Longest Common Ending.

```
using System;
Public class Program
{
    Public static void Main()
    {
        Console.WriteLine(LongestCommonEditing("multiplication","ration"));
        Console.WriteLine(LongestCommonEditing("potent","tent"));
        Console.WriteLine(LongestCommonEditing("SkyScraper","carnivore"));
        Console.ReadKey();
    }
    Public static string longestCommonEnding(string s1, string s2);
    {
```

```
        int i = S1.Length - 1;
        int j = S2.Length - 1;
        string CommonEnding = "";
        while (i >= 0 && j >= 0 && S1[i] == S2[j])
        {
            CommonEnding = S1[i] + CommonEnding;
            i--;
            j--;
        }
        return CommonEnding;
    }
}
```

Output:

action
tent

3.7 Clear Brackets

```
using System;
using System.Collections.Generic;
Public class Program
{
    Public static void Main()
    {
        Console.WriteLine(Brackets("(a*(b-c)-...)"));
        Console.WriteLine(Brackets("(a-b-15)*(a-34)"));
        Console.WriteLine(Brakets("sin(a)-...)+---- cos1)"));
        Console.ReadLine();
    }
    Public static bool Brackets(string expression)
    {
        Stack<char> stack = new Stack<char>();
        foreach (char ch in expression)
        {
            if (ch == '_' || !IsBracket(ch))
                continue;
            if (ch == '(')
```

```
        Stack.Push(ch);
    }
    else if (ch == ')')
    {
        if (Stack.Count == 0)
        {
            return false;
        }
        Stack.Pop();
    }
    return Stack.Count == 0;
}
Private static bool IsBracket(char ch)
{
    return ch == '(' || ch == ')';
}
}
```

Output
True
False
False

3.8 Count the no. of duplicate characters

```
Using System;
Using System.Collections.Generic;
Public class program
{
    Public static void main()
    {
        Console.WriteLine(Duplicate Count("abcde"));
        Console.WriteLine(Duplicate Count("aabbcde"));
        Console.WriteLine(Duplicate Count("Indivisibilities"));
        Console.WriteLine(Duplicate count("Aa"));
        Console.ReadKey();
    }
```

ROLL NUMBER :

```
Public static int DuplicateCount (string input)
{
 Dictionary < char, int > charCount, new dictionary
< char, int > (); foreach (char ch in input)
  {
    if (char. IsLetter or Digit (ch))
     {
       If (charCount. Contains key (ch))
        {
          charCount [ch] ++;
        }
        else
        {
          charCount [ch]: 1;
        }
     }
  }
    int duplicateCount = 0;
  foreach (int count in charCount values)
    {
      if (count >1)
       {
         duplicateCount ++;
       }
      return duplicate
    }
}
```

output

0
2
2
0

ROLL NUMBER :

3.9 Uban Numbers

```
Using system;
Public class program
{
    Public static void main () {
        Console. write line (is uban (456));
        Console . write line (is ~~tot~~ uban (25));
        Console . write line (is uban (1098));
        Console . write line (is uban (1000000));
        Console . Read Key ();
    }
    Public static bool Isuban (int number)
    {
        String word = Number Towards (number);
        return ! word . Contains ("U");
    }
    Private static string Number towards (int number)
    {
        if (number == 0)
        return "zero";
        if (number < 0)
    Throw new Argument Exception ("Number must be non-negative");
    string [ ] below twenty = {"zero", "one", "two", "three", "four",
    "five", "six", "seven", "eight", "nine", "ten", "eleven", "twelve"
    "thirteen", "fourteen", "fifteen", "sixteen", "seventeen",
    "eighteen", "nineteen" };
string [ ] tens = {" ", " ", "twenty", "thirty", "fourty", "fifty",
    "sixty", "seventy", "eighty", "ninety" };
string [ ] thousands, {" ", "thousand", "million", "billion" };
```

```
String words(int n)
{
 if (n < 20)
 return below Twenty[n];
 if (n<100)
 return tens[n/10] + (n%10>0 ? " " + below twenty[n%10]; "");
 if (n<1000)
 return below Twenty[n/100] + " hundred" + (n%100>0 ? " and
                " + words(n%100) : "");
 for (int i=0; i< thousands.length; i++)
 {
 int divisor = (int) math.pow(1000, i++);
 if (n < divisor)
 return words n/(divisor/1000)) + " " + thousands[i] +
        (n%(divisor/1000)>0 ? " " + words(n%divisor/
                                1000) : "");
    }
    Throw new Argument out of Range Exception();
 }
 return words(number).Trim();
 }
}
```

Out put

false
True
false
True