**Name: Muhammad Awais Shaikh**

**Student ID: 20042**

**Course: Data Structure And Algorithms Lab**

**Submitted To: Sir Faraz Abdul Basit**

## Task #1

//Print name namespace HelloWorld

{

# (C SHARP - C#) LAB 1

class Program

{

static void Main(string[] args)

//Print name

{

Console.WriteLine("Hello World!"); Console.Read();

}

}

}­­

## Task#2

namespace HelloWorld

{

class Program

{

static void Main(string[] args)

{

//Variable - string

string name = "My name is Awais shaikh"; Console.WriteLine(name); Console.Read();

//Variable - int int myNum = 15;

Console.WriteLine(myNum);

Console.Read();

//Variable - double

double myDoubleNum = 3.39D; Console.WriteLine(myDoubleNum); Console.Read();

//Variable - char char myLetter = 'A';

Console.WriteLine(myLetter); Console.Read();

//Variable - float

float myfloatNum = 6.99F; Console.WriteLine(myfloatNum); Console.Read();

}

}

## Task#3

namespace HelloWorld

{

class Program

{

static void Main(string[] args)

{

//User Input Console.WriteLine("Enter your age:");

int age = Convert.ToInt32(Console.ReadLine()); Console.WriteLine("Your age is: " + age); Console.Read();

}

}

}

## Task#4

namespace ConsoleApplication2

{

class Program

{

static void Main(string[] args)

{

string[] names = new string[Length]; for(int i=0; i<=Length; i++)

{

names[i] = Console.Read();

}

OR

Console.WriteLine("enter array length");

int length = Convert.ToInt32(Console.ReadLine()); string[] names = new string[length];

for (int i = 0; i < length; i++)

{

names[i] = Console.ReadLine();

}

}

}

}

# LAB 2

## Task#1

namespace HelloWorld

{

class Program

{

static void Main(string[] args)

{

//Concatenation Console.WriteLine("Enter your age:");

int age = Convert.ToInt32(Console.ReadLine()); Console.WriteLine("Your age is: " + age);

//Length

string text = "Germany";

Console.WriteLine("The Length of Germany is: " + text.Length);

//Extracting the Substring string a = "Awais";

Console.WriteLine(a.Substring(2)); Console.Read();

}

}

}

# LAB 3

## Task #1

//Storing values through array

namespace ConsoleApplication2

{

class Program

{

static void Main(string[] args)

{

int[] myNum = { 10, 20, 30, 40, 50, 60};

Console.WriteLine(myNum[3]); Console.Read();

}

}

}

## Task #2

**//Taking user input in array without looping**

namespace ConsoleApplication2

{

class Program

{

static void Main(string[] args)

{

string[] names = new string [5]; Console.WriteLine("\*\*\*\*\*\*Input\*\*\*\*\*\*");

Console.WriteLine("Enter name at 0 index:"); names[0] = Console.ReadLine();

Console.WriteLine("Enter name at 1 index:"); names[1] = Console.ReadLine();

Console.WriteLine("Enter name at 2 index:"); names[2] = Console.ReadLine();

Console.WriteLine("\*\*\*\*\*\*Display\*\*\*\*\*\*");

Console.WriteLine("Enter name at 0 index:" +names[0]); Console.WriteLine("Enter name at 1 index:" +names[1]); Console.WriteLine("Enter name at 2 index:" +names[2]);

Console.Read();

}

}

}

# LAB 4

## Task #2

**//2d array without looping**

using System;

class Program

{

static void Main()

{

string[][] users = new string[4][];

users[0] = new string[4]; users[1] = new string[4]; users[2] = new string[4]; users[3] = new string[4];

users[0][0] = "S.NO";

users[0][1] = "Names";

users[0][2] = "ID'S";

users[0][3] = "Contact";

users[1][0] = "1";

users[1][1] = "Awais";

users[1][2] = "23003";

users[1][3] = "03247900621";

users[2][0] = "2";

users[2][1] = "uzma";

users[2][2] = "22429";

users[2][3] = "02347485931";

users[3][0] = "3";

users[3][1] = "Hamza";

users[3][2] = "22429";

users[3][3] = "02462904177";

// Concatenate and print the values horizontally

Console.WriteLine(string.Join(", ", users[0])); Console.WriteLine(string.Join(", ", users[1])); Console.WriteLine(string.Join(", ", users[2])); Console.WriteLine(string.Join(", ", users[3])); Console.Read();

}

}

## Task#3

**//2d array using loop**

using System;

class Program

{

static void Main()

{

string[][] users = new string[4][];

users[0] = new string[4]; users[1] = new string[4]; users[2] = new string[4]; users[3] = new string[4];

users[0][0] = "S.NO";

users[0][1] = "Names";

users[0][2] = "ID'S";

users[0][3] = " Contact";

users[1][0] = " 1";

users[1][1] = " Awais";

users[1][2] = "23003";

users[1][3] = "03247900621";

users[2][0] = " 2";

users[2][1] = " Aliza";

users[2][2] = "22429";

users[2][3] = "02347485931";

users[3][0] = " 3";

users[3][1] = " Hamza";

users[3][2] = "22429";

users[3][3] = "02462904177";

for (int i = 0; i < users.Length; i++)

{

string line = string.Join(" ", users[i]); Console.WriteLine(line);

}

Console.Read();

}

}

## Task#4

using System; class Program

{

static void Main()

{

string[][] users = new string[4][]; users[0] = new string[4]; users[1] = new string[4]; users[2] = new string[4]; users[3] = new string[4];

users[0][0] = "ID";

users[0][1] = "NAME";

users[0][2] = "EMAIL";

users[0][3] = "password";

users[1][0] = "2282";

users[1][1] = "Awais"; users[1][2] = "[awais@gmail.com](mailto:awais@gmail.com)"; users[1][3] = "123";

users[2][0] = "2282";

users[2][1] = "Awais"; users[2][2] = "[awais@gmail.com](mailto:awais@gmail.com)"; users[2][3] = "123";

users[3][0] = "2282";

users[3][1] = "Awais"; users[3][2] = "[awais@gmail.com](mailto:awais@gmail.com)"; users[3][3] = "123";

Console.WriteLine("User Data:");

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

Console.Write(users[i][j] + "\t");

}

Console.WriteLine();

}

Console.Read();

}

}

## Task#4

**//taking input from user**

using System; class Program

{

static void Main()

{

string[][] users = new string[4][];

users[0] = new string[4]; users[1] = new string[4]; users[2] = new string[4]; users[3] = new string[4];

for (int i = 1; i < users.Length; i++)

{

Console.WriteLine("Enter data for user {i}:");

Console.Write("S.NO: "); users[i][0] = Console.ReadLine();

Console.Write("Names: "); users[i][1] = Console.ReadLine();

Console.Write("ID'S: "); users[i][2] = Console.ReadLine();

Console.Write("Contact: "); users[i][3] = Console.ReadLine();

Console.WriteLine();

}

string header = string.Join(" ", users[0]); Console.WriteLine(header);

for (int i = 1; i < users.Length; i++)

{

string line = string.Join(" ", users[i]); Console.WriteLine(line);

}

Console.Read();

}

}

## Task#5

**// determine the size of the users array**

using System; class Program

{

static void Main()

{

Console.Write("Enter the number of users: ");

int numberOfUsers = int.Parse(Console.ReadLine());

// Create a 2D array to store user data string[][] users = new string[numberOfUsers][];

// Prompt the user to enter data for each user for (int i = 0; i < numberOfUsers; i++)

{

Console.WriteLine($"Enter data for user {i + 1}:");

Console.Write("S.NO: ");

string sno = Console.ReadLine();

Console.Write("Names: ");

string name = Console.ReadLine();

Console.Write("ID'S: ");

string id = Console.ReadLine();

Console.Write("Contact: ");

string contact = Console.ReadLine();

// Create an array to store the user data users[i] = new string[] { sno, name, id, contact };

Console.WriteLine();

}

// Display the header

string header = string.Join(" ", "S.NO", "Names", "ID'S", "Contact"); Console.WriteLine(header);

// Display user data

for (int i = 0; i < numberOfUsers; i++)

{

string line = string.Join(" ", users[i]); Console.WriteLine(line);

}

Console.Read();

}

}

## PUSH And POP

using System;

using System.Collections;

namespace DemoApplication

{

class Program

{

static void Main(string[] args)

{

Stack stackVar = new Stack(); stackVar.Push(1); stackVar.Push(2); stackVar.Push(3);

stackVar.Pop(); stackVar.Pop();

foreach (var storeValue in stackVar)

{

Console.WriteLine(storeValue);

}

Console.Read();

}

}

}

# PUST, POP, COUNT AND PEEK METHOD

using System.Collections;

namespace ConsoleApp1

{

internal class Program

{

static void Main(string[] args)

{

//push -> add method Stack obj = new Stack(); obj.Push(1);

obj.Push(2);

obj.Push(3);

obj.Push(4);

obj.Push(5);

//top of the value in stack, using peek method Console.WriteLine(" Using peek method");

Console.WriteLine("Top of the value in stack : " + obj.Peek() + "\n");

"\n");

"\n");

//before pop method , calculate total elements , using count method Console.WriteLine("Using count method");

Console.WriteLine("Before, total elements are calculate : " + obj.Count +

//pop -> delete method Console.WriteLine("Using pop method"); obj.Pop();

//after pop method , calculate total elements , using count method Console.WriteLine("After, total elements are calculate : " + obj.Count +

// using loop

Console.WriteLine("calculate total value in stack ");

foreach (int store\_box in obj)

{

Console.WriteLine("push elements " +store\_box);

}

Console.Read();

}

}

}

# LAB 5

## Task 1

**Using queue enqueue and dequeue data and print**

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LabManual

{

internal class QueueWork

{

public static void AddandDisplayQueueDAta() {

var Names = new Queue<string>();

Names.Enqueue(Console.ReadLine()); Names.Enqueue(Console.ReadLine()); Names.Enqueue(Console.ReadLine());

foreach (var name in Names)

{

Console.WriteLine(name);

}

Names.Dequeue();

Console.WriteLine("The peaked item is " + Names.Peek()); Console.WriteLine("The deleted item is " + Names.Dequeue()); Console.WriteLine("The peaked item is " + Names.Peek());

Console.ReadLine();

}

}

}

# LAB 6

**Task 1**

# LINEAR SEARCH

using System;

namespace LabManual

{

internal class Program

{

static void Main(string[] args)

{

int[] array = { 1, 2, 3, 4, 5 }; int? index = null;

Console.WriteLine("Enter a value for search"); int b = Convert.ToInt32(Console.ReadLine());

for (int i = 0; i < array.Length; i++)

{

if (b == array[i])

{

index = i; break;

}

}

if (index != null)

{

Console.Write("Found value at index: " + index);

}

else

{

Console.Write("Not Found");

}

Console.ReadLine();

}

}

}

# LAB 7

## Task 1

**Code of linear search**

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LabManual

{

internal class Program

{

static void Main(string[] args)

{

int key; Console.WriteLine("Enter Key");

key = Convert.ToInt32(Console.ReadLine());

//int[] arr = { 10, 20, 30, 40, 50, 69 };

binarySearch.binarymethod(key); Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace binarySearch

{

internal class binarySearch

{

public static int binarymethod(int key )

{

int[] arr1 = new int[5]; for (int i = 0; i < 5; i++)

{

Console.WriteLine("ENTER value AT : " + i); arr1[i] = Convert.ToInt32(Console.ReadLine());

}

int min = 0;

int max = arr1.Length - 1; while (min <= max)

{

int mid = (min + max) / 2; if (key == arr1[mid])

{

Console.WriteLine("Found element at " + mid + " " + "index"); return ++mid;

}

else if (key < arr1[mid])

{

max = mid - 1;

}

else

{

min = mid + 1;

}

}

Console.WriteLine("key dosent found"); return 0;

}

}

# LAB 8

## Task 1

**Code of binary search**

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

using System.Threading.Tasks; namespace LabManual

{

internal class Program

{

static void Main(string[] args)

{

Console.Write("Enter the size of the array: "); int size = int.Parse(Console.ReadLine());

int[] arr = new int[size];

Console.WriteLine("Enter sorted array elements:"); for (int i = 0; i < size; i++)

{

arr[i] = int.Parse(Console.ReadLine());

}

Console.Write("Enter the key to search for: "); int key = int.Parse(Console.ReadLine());

int result = BinarySearchClass.BinarySearchMethod(arr, key);

if (result != -1)

{

Console.WriteLine("Key found at index: " + result);

}

else

{

Console.WriteLine("Key not found in the array.");

}

Console.ReadLine();

}

}

public class BinarySearchClass

{

public static int BinarySearchMethod(int[] inputArray, int key)

{

int min = 0;

int max = inputArray.Length - 1; while (min <= max)

{

int mid = (min + max) / 2; if (key == inputArray[mid])

{

return mid;

}

else if (key < inputArray[mid])

{

max = mid - 1;

}

else

{

min = mid + 1;

}

}

return -1; // Key not found

}

}

}

# LAB 9

## Task 1

**Code of bubble sort**

using System;

namespace LabManual

{

internal class Program

{

static void Main(string[] args)

{

int[] arr = { 11, 212, 33, 412, 512, 643, 712, 82 };

for (int i = 0; i < arr.Length; i++)

{

Console.WriteLine(" " + arr[i]);

}

Console.WriteLine(); Console.WriteLine();

Console.WriteLine("Enter search value from above Array"); int target = Convert.ToInt32(Console.ReadLine());

int result = Array.Find(arr, element => element == target); if (result != 0)

{

Console.WriteLine("Element found at index: " + Array.IndexOf(arr,

result));

}

else

{

Console.WriteLine("Element not found in the array.");

}

Console.ReadLine();

}

}

# LAB 10

**Task 1**

# LINEAR SEARCH USING CLASS

using System;

namespace LabManual

{

internal class Program

{

static void Main(string[] args)

{

search.L\_search();

}

}

}

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LINEAR\_SEARCH

{

internal class search

{

public static void L\_search()

{

int[] array = { 1, 2, 3, 4, 5 }; int? index = null;

Console.WriteLine("Enter a value for search"); int b = Convert.ToInt32(Console.ReadLine());

for (int i = 0; i < array.Length; i++)

{

if (b == array[i])

{

index = i; break;

}

}

if (index != null)

{

Console.Write("Found value at index: " + index);

}

else

{

Console.Write("Not Found");

}

Console.ReadLine();

}

}

# LAB 11

## Task 1

***QUIZ***

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

using System.Threading.Tasks; namespace QuizAssignment

{

class Program

{

static void Main(string[] args)

{

University university = new University();

university.AddDepartment(new Department(3, "Computer Science", "Sir Zubair", 250));

university.AddDepartment(new Department(1, "Physics", "Sir Zubair", 120));

university.AddDepartment(new Department(5, "History", "Sir Sabeeh", 180));

university.AddDepartment(new Department(2, "Mathematics", "Sir Faraz", 200));

university.AddDepartment(new Department(4, "Biology", "Sir Zamin", 150));

Console.WriteLine("Before Sorting:"); university.DisplayDepartments(); university.SortDepartments(); Console.WriteLine("\nAfter Sorting:"); university.DisplayDepartments(); Console.Write("\nEnter Department ID to search: "); int searchId = int.Parse(Console.ReadLine());

Department foundById = university.SearchDepartmentById(searchId);

if (foundById != null)

{

Console.WriteLine("\nSearch Result (by ID):\n" + foundById);

}

else

{

Console.WriteLine("\nDepartment not found.");

}

Console.Write("\nEnter Department Name to search: "); string searchName = Console.ReadLine();

Department foundByName = university.SearchDepartmentByName(searchName);

if (foundByName != null)

{

Console.WriteLine("\nSearch Result (by Name):\n" + foundByName);

}

else

{

Console.WriteLine("\nDepartment not found.");

}

Console.ReadLine();

}

}

}

# LAB 12

## Task 1 TREE

using System; using System.Linq; using System.Text;

using System.Threading.Tasks;

namespace Trees

{

using System.Collections.Generic; internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Awais Khan., 20042"); TreeStructure.root = new Node(10); TreeStructure.root.left = new Node(11); TreeStructure.root.left.left = new Node(7); TreeStructure.root.right = new Node(9); TreeStructure.root.right.left = new Node(15); TreeStructure.root.right.right = new Node(8); Console.Write("Inorder traversal before insertion: "); TreeStructure.inorder(TreeStructure.root);

int key = 12; TreeStructure.insert(TreeStructure.root, key);

Console.Write("\nInorder traversal after insertion: "); TreeStructure.inorder(TreeStructure.root);

int deleteKey = 10; TreeStructure.delete(TreeStructure.root, deleteKey); Console.Write("\nInorder traversal " + "after deletion: "); TreeStructure.inorder(TreeStructure.root); Console.ReadKey();

}

}

}

using System;

using System.Collections.Generic; using System.Linq;

using System.Text;

using System.Threading.Tasks; using System.Xml.Linq;

namespace Trees

{

public class Node

{

public int key;

public Node left, right;

// constructor public Node(int key)

{

this.key = key; left = null; right = null;

}

}

}

using System;

using System.Collections.Generic; namespace Trees

{

public class TreeStructure

{

public static Node root;

public static void inorder(Node temp)

{

if (temp == null) return;

inorder(temp.left); Console.Write(temp.key + " "); inorder(temp.right);

}

//function to insert element in binary tree public static void insert(Node temp, int key)

{

if (temp == null)

{

root = new Node(key); return;

}

Queue<Node> q = new Queue<Node>(); q.Enqueue(temp);

// Do level order traversal until we find

// an empty place. while (q.Count != 0)

{

temp = q.Peek(); q.Dequeue();

if (temp.left == null)

{

temp.left = new Node(key); break;

}

else

q.Enqueue(temp.left);

if (temp.right == null)

{

temp.right = new Node(key); break;

}

else

q.Enqueue(temp.right);

}

}

//function to Delete element in binary tree

static void deleteDeepest(Node root, Node delNode)

{

Queue<Node> q = new Queue<Node>();

q.Enqueue(root); Node temp = null;

// Do level order traversal until last node while (q.Count != 0)

{

temp = q.Peek(); q.Dequeue();

if (temp == delNode)

{

temp = null; return;

}

if (temp.right != null)

{

if (temp.right == delNode)

{

temp.right = null; return;

}

else

q.Enqueue(temp.right);

}

if (temp.left != null)

{

if (temp.left == delNode)

{

temp.left = null; return;

}

else

q.Enqueue(temp.left);

}

}

}

public static void delete(Node root, int key)

{

if (root == null) return;

if (root.left == null && root.right == null)

{

if (root.key == key)

{

root = null; return;

}

else

return;

}

Queue<Node> q = new Queue<Node>(); q.Enqueue(root);

Node temp = null, keyNode = null;

// Do level order traversal until

// we find key and last node. while (q.Count != 0)

{

temp = q.Peek(); q.Dequeue();

if (temp.key == key) keyNode = temp;

if (temp.left != null) q.Enqueue(temp.left);

if (temp.right != null) q.Enqueue(temp.right);

}

if (keyNode != null)

{

int x = temp.key; deleteDeepest(root, temp); keyNode.key = x;

}

}

}

}