***Mobile Application Development Lab***

***CSL-341***

***Lab Journal***



**Student Name: Syed Amber Ali Shah**

**Enrollment No: 01-134222-146**

**Class and Section: BSCS 6-B**

**Department of Computer Science**

**BAHRIA UNIVERSITY ISLAMABAD**

# Lab 2 – Dart Introduction

## Objectives:

Basic syntax of Dart programming language

## Tools Used:

VS Code

### Submission Date:

## Evaluation Signatures of Lab Instructor

# TASK 1:

Find the largest number in a given list.

**Solution:**

void main()

{

  List<int> numbers = [-15,15,0,50,49];

  int max = grt(numbers);

  print("The greatest number is: $max");

}

int grt(List<int> n)

{

  int max = n[0];

  for (int i in n)

  {

    if (i > max) {

      max = i;

    }

  }

  return max;

**Output:**

****

# TASK 2:

Use merge sort to sort a List.

**Solution:**

void merge(List<int> arr, int left, int mid, int right) {

  int n1 = mid - left + 1;

  int n2 = right - mid;

  List<int> leftArr = List<int>.filled(n1, 0);

  List<int> rightArr = List<int>.filled(n2, 0);

  for (int i = 0; i < n1; i++) {

    leftArr[i] = arr[left + i];

  }

  for (int j = 0; j < n2; j++) {

    rightArr[j] = arr[mid + 1 + j];

  }

  int i = 0, j = 0, k = left;

  while (i < n1 && j < n2) {

    if (leftArr[i] <= rightArr[j]) {

      arr[k] = leftArr[i];

      i++;

    } else {

      arr[k] = rightArr[j];

      j++;

    }

    k++;

  }

  while (i < n1) {

    arr[k] = leftArr[i];

    i++;

    k++;

  }

  while (j < n2) {

    arr[k] = rightArr[j];

    j++;

    k++;

  }

}

// Merge Sort function

void mergeSort(List<int> arr, int left, int right) {

  if (left < right) {

    int mid = left + (right - left) ~/ 2;

    mergeSort(arr, left, mid);

    mergeSort(arr, mid + 1, right);

    merge(arr, left, mid, right);

  }

}

// Function to print the array

void printArray(List<int> arr) {

  print(arr.join(" "));

}

// Main function

void main() {

  List<int> arr = [12, 11, 13, 5, 6, 7];

  print("Original array: ");

  printArray(arr);

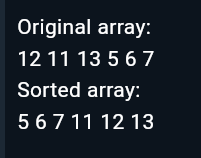
  mergeSort(arr, 0, arr.length - 1);

  print("Sorted array: ");

  printArray(arr);

}

**Output:**



# Task 3:

Implement a Stack from Scratch.

**Solution:**

class Stack<T> {

  List<T> \_stack = [];

  void push(T value) {

    \_stack.add(value);

  }

  T? pop() {

    if (isEmpty) {

      print("Stack is empty!");

      return null;

    }

    return \_stack.removeLast();

  }

  T? peek() {

    if (isEmpty) {

      print("Stack is empty!");

      return null;

    }

    return \_stack.last;

  }

  // Check if the stack is empty

  bool get isEmpty => \_stack.isEmpty;

  // Get the size of the stack

  int get size => \_stack.length;

  // Print the stack

  void display() {

    print(\_stack.reversed.toList());

  }

}

void main() {

  Stack<int> stack = Stack<int>();

  stack.push(10);

  stack.push(20);

  stack.push(30);

  print("Stack after pushing 10, 20, 30:");

  stack.display();

  print("Top element: ${stack.peek()}");

  print("Popped element: ${stack.pop()}");

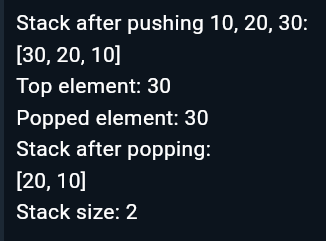
  print("Stack after popping:");

  stack.display();

  print("Stack size: ${stack.size}");

}

**Output:**



**THE END**