

LAB NO 4

DATA STRUCTURES AND ALGORITHMS

OBJECTIVE: To understand arrays and its memory allocation.

LAB TASKS

1. Write a program that takes two arrays of size 4 and swap the elements of those arrays.

Input 01:

```
package lab.no.pkg4;
import java.util.Arrays;
public class LABNO4 {

    public static void main(String[] args) {

        // Write a program that takes two arrays of size 4 and swap the elements of those arrays.

        int a1[] = {2,4,6,8};
        int a2[] = {1,3,5,7};

        System.out.println("Before swap array 1 is: " + Arrays.toString(a1));
        System.out.println("Before swap array 2 is: " + Arrays.toString(a2));

        int temp [] = a1;
        a1 = a2;
        a2 = temp;
        System.out.println();
        System.out.println("After swap array 1 is: " + Arrays.toString(a1));
        System.out.println("After swap array 2 is: " + Arrays.toString(a2));
    }
}
```

Output 01:

```
Output - LAB NO 4 (run)

run:
Before swap array 1 is: [2, 4, 6, 8]
Before swap array 2 is: [1, 3, 5, 7]

After swap array 1 is: [1, 3, 5, 7]
After swap array 2 is: [2, 4, 6, 8]
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

2. Add a method in the class that takes array and merge it with the existing one.

Input 02:

```
public static void merge_Array(int a1[] ,int a2[] ){  
  
    int a3[] = new int [a1.length + a2.length];  
  
    for(int i = 0 ; i < a1.length; i++ ){  
        a3[i] = a1[i];  
    }  
  
    for(int i = 0; i < a2.length; i++){  
        a3[a1.length + i] = a2[i];  
    }  
    System.out.println("The merged array (a3) is : " + Arrays.toString(a3) );  
}  
  
public static void main(String[] args) {  
  
    int[] a1 = {15, 16, 17, 18};  
    int[] a2 = {19, 20, 21, 22};  
  
    merge_Array(a1,a2);  
}
```

Output 02:

```
Output - LAB NO 4 (run)  
  
run:  
The merged array (a3) is : [15, 16, 17, 18, 19, 20, 21, 22]  
BUILD SUCCESSFUL (total time: 0 seconds)  
|
```

3. In a JAVA program, take an array of type string and then check whether the strings are palindrome or not.

Input 03:

```
public static void palindrome_checker(String a1[]){  
  
    String [] a2 = new String [a1.length];  
    System.out.println("The original array is : " + Arrays.toString(a1));  
  
    for(int i = 0; i < a1.length ; i++){  
        a2[i] = new StringBuilder( a1[i] ).reverse().toString();  
    }  
  
    System.out.println(" The array with reversed String is :"+ Arrays.toString(a2));  
  
    for (int i = 0; i < a1.length; i++) {  
        if (a1[i].equalsIgnoreCase(a2[i])) {  
            System.out.println(a1[i] + " is a palindrome.");  
        }  
        else {  
            System.out.println(a1[i] + " is not a palindrome.");  
        }  
    }  
}  
  
public static void main(String[] args) {  
    String a1[] = {"WAQAR" , "WOW" , "RIASAT" ,"LOL","SARAH" ,"MOM"};  
    palindrome_checker(a1);  
}
```

Output 03:

```
Output - LAB NO 4 (run)  
  
run:  
The original array is : [WAQAR, WOW, RIASAT, LOL, SARAH, MOM]  
The array with reversed String is :[RAQAW, WOW, TASAIR, LOL, HARAS, MOM]  
WAQAR is not a palindrome.  
WOW is a palindrome.  
RIASAT is not a palindrome.  
LOL is a palindrome.  
SARAH is not a palindrome.  
MOM is a palindrome.  
BUILD SUCCESSFUL (total time: 0 seconds)  
|
```

4. Given an array of integers, count how many numbers are even and how many are odd.

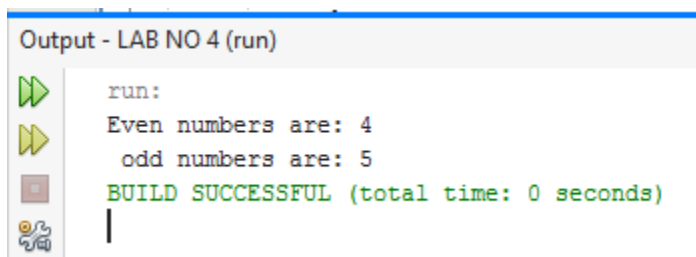
Input 04:

```
// Given an array of integers, count how many numbers are even and how many are odd.
public static void even_odd_count(int arr[]){
    int count_even = 0;
    int count_odd = 0;

    for(int i = 0; i < arr.length; i++){
        if(arr[i] % 2 == 0 ){
            count_even+=1;
        }
        else{
            count_odd+= 1;
        }
    }
    System.out.println("Even numbers are: " + count_even + "\n odd numbers are: " + count_odd);
}

public static void main(String[] args) {
    int arr[] = {1,2,3,4,5,6,7,8,9};
    even_odd_count(arr);
}
```

Output 04:



Output - LAB NO 4 (run)

run:
Even numbers are: 4
odd numbers are: 5
BUILD SUCCESSFUL (total time: 0 seconds)

5. Given two integer arrays, merge them and remove any duplicate values from the resulting array

Input 05:

```
public static void merge_Array(int a1[] ,int a2[] ){

    System.out.println("array a1 " + Arrays.toString(a1));
    System.out.println("array a2 " + Arrays.toString(a2));

    int a3[] = new int [a1.length + a2.length];

    for(int i = 0 ; i < a1.length; i++ ){
        a3[i] = a1[i];
    }

    for(int i = 0; i < a2.length; i++){
        a3[a1.length + i] = a2[i];
    }

    System.out.println("The merged array (a3) is : " + Arrays.toString(a3) );

    int[] tempArray = new int[a3.length];
    int tempIndex = 0;

    for (int i = 0; i < a3.length; i++) {
        boolean isDuplicate = false;

        for (int j = 0; j < tempIndex; j++) {
            if (a3[i] == tempArray[j]) {
                isDuplicate = true;
                break;
            }
        }

        if (!isDuplicate) {
            tempArray[tempIndex++] = a3[i];
        }
    }

    int[] resultArray = Arrays.copyOf(tempArray, tempIndex);

    System.out.println("Merged array without duplicates: " + Arrays.toString(resultArray));
}

public static void main(String[] args) {

    int[] a1 = {15, 16, 17, 18};
    int[] a2 = {19, 16, 21, 18};

    merge_Array(a1,a2);
}
```

Output 05:

```
Output - LAB NO 4 (run)

run:
array a1 [15, 16, 17, 18]
array a2 [19, 16, 21, 18]
The merged array (a3) is : [15, 16, 17, 18, 19, 16, 21, 18]
Merged array without duplicates: [15, 16, 17, 18, 19, 21]
BUILD SUCCESSFUL (total time: 0 seconds)
```

HOME TASKS

1. Write a program that takes an array of Real numbers having size 7 and calculate the sum and mean of all the elements. Also depict the memory management of this task.

Input 01:

```
public static void main(String[] args) {
    int real_num [] = {1,2,3,4,5,6,7};
    System.out.println("real numbers are: " + Arrays.toString(real_num));
    int sum = 0;
    for(int num:real_num){
        sum+= num;
    }
    System.out.println("the sum of array having real numbers is: " + sum);

    int n = real_num.length;
    int mean = sum / n;
    System.out.println("the mean of the array is: " + mean);
}
```

Output 01:

```
Output - LAB NO 4 (run)

run:
real numbers are: [1, 2, 3, 4, 5, 6, 7]
the sum of array having real numbers is: 28
the mean of the array is: 4
BUILD SUCCESSFUL (total time: 0 seconds)
```

Memory Usage

- **Heap Memory:**
 - `real_num []` : occupies 28 bytes because of (7 integers) , each integer occupying 4 bytes.
 - `Arrays.toString(real_num)`: depending on the string length. a new temporary array (or string) is created in memory to hold the string representation of the array. This will also consume some memory in the heap
- **Stack Memory:**
 - `sum`: 4 bytes (single integer sum variable is allocated in the stack memory)
 - `n`: 4 bytes (allocated in the stack memory holding the length of array.)
 - `mean`: 4 bytes (allocated in the stack frame to store the calculated mean)
 - `num`: 4 bytes (reused during loop iterations)

Total Memory Usage

- Total Stack Memory: $4 (\text{sum}) + 4 (n) + 4 (\text{mean}) + 4 (\text{num}) = 16$ bytes
- Total Heap Memory: 28 bytes (for the array) + temporary string size.

Garbage Collection:

- Once the main method completes execution, the Java garbage collector will automatically reclaim the memory used for the `real_num` array and any temporary objects created during the execution of the method. The stack memory will be released as the stack frame for main is popped off the stack

2. Add a method in the same class that splits the existing array into two. The method should search a key in array and if found splits the array from that index of the key

Input 02:

```
public static void search_split(int arr1[], int key){

    int index = -1;
    System.out.println("The key is: " + key);
    System.out.println(" The array is: " + Arrays.toString(arr1));
    System.out.println();

    for(int i = 0 ; i< arr1.length; i++){

        if(arr1[i] == key ){
            index = i;
            System.out.println("key " + key + " is found at index " + index);
            break;
        }
    }

    if(index == -1){
        System.out.println("Invalid key / key not found");
    }

    else{
        int a1[] = Arrays.copyOfRange(arr1, 0, index);
        int a2[] = Arrays.copyOfRange(arr1, index, arr1.length);

        System.out.println("The first splitted part from index 0 to key " + key + " is " + Arrays.toString(a1));
        System.out.println("The second splitted part from key " + key + " to index " + arr1.length + " is " + Arrays.toString(a2))
    }
}

public static void main(String[] args) {
    int arr1[] = {19,18,17,16,15,14,13,12,11,10};
    int k = 14;
    search_split(arr1,k);
}
```

Output 02:

```
Output - LAB NO 4 (run)

run:
The key is: 14
The array is: [19, 18, 17, 16, 15, 14, 13, 12, 11, 10]

key 14 is found at index 5
The first splitted part from index 0 to key 14 is [19, 18, 17, 16, 15]
The second splitted part from key 14 to index 10 is [14, 13, 12, 11, 10]
BUILD SUCCESSFUL (total time: 0 seconds)
```


3. Given an array of distinct integers and a target integer, return all unique combinations of numbers that add up to the target. Each number can be used only once in the combination.

Input 03:

```
public static void Combinations_Finder(int[] disc_int, int t) {
    int[] combination = new int[disc_int.length]; // combinations ko store krwayega
    backtrack(disc_int, t, 0, combination, 0); // method to find all the combinations that sum up to the target t.
}

public static void backtrack(int[] disc_int, int t, int start, int[] combination, int index) {
    // check karega agar target achieve hogya to combinations ko print krdega ..
    if (t == 0) {
        for (int i = 0; i < index; i++) {
            System.out.print(combination[i] + " ");
        }
        System.out.println();
        return;
    }
    // agar distinct integers me koi value target se greater hogi to osko skip krdo.
    for (int i = start; i < disc_int.length; i++) { // or rest of the values ko combinations[] me add kro
        if (disc_int[i] > t) continue;
        combination[index] = disc_int[i];
        backtrack(disc_int, t - disc_int[i], i + 1, combination, index + 1); // Recur with reduced target
    }
}

public static void main(String[] args) {
    int[] disc_int = {1, 3, 6, 2, 7, 8};
    System.out.println("the distinct integers array is: " + Arrays.toString(disc_int));
    int t = 9;
    System.out.println("the Target is: " + t);
    Combinations_Finder(disc_int, t);
}
```

Output 03:

```
Output - LAB NO 4 (run)

run:
the distinct integers array is: [1, 3, 6, 2, 7, 8]
the Target is: 9
1 6 2
1 8
3 6
2 7
BUILD SUCCESSFUL (total time: 1 second)
```

4. You are given an array containing n distinct numbers taken from $0, 1, 2, \dots, n$. Write a program to find the one number that is missing from the array.

Input 04:

```
// You are given an array containing n distinct numbers taken from 0, 1, 2, ..., n.
// Write a program to find the one number that is missing from the array
public static int missing(int a1[]){

    int sum_a1 = 0 ;

    int n = a1.length +1;
    System.out.println("length of a1 is " + n);
    int sum_n = n*(n+1) / 2;
    System.out.println("sum to n is " +sum_n);

    for(int num : a1 ){
        sum_a1 += num;
    }

    System.out.println("sum of a1 is "+sum_a1);
    return sum_n - sum_a1;
}

public static void main(String[] args) {
    int a1[] = {1,3,4,5,6,7,8,9,10};
    System.out.println("The missing number is: " + missing(a1));
}
```

Output 04:

```
Output - LAB NO 4 (run)

run:
length of a1 is 10
sum to n is 55
sum of a1 is 53
The missing number is: 2
BUILD SUCCESSFUL (total time: 0 seconds)
|
```

5. You are given an array of integers. Write a program to sort the array such that it follows a zigzag pattern: the first element is less than the second, the second is greater than the third, and so on.

Input 05:

```
public static void Zig_ZAG(int[] a1) {
    for (int i = 0; i < a1.length - 1; i++) {
        if (i % 2 == 0) {
            if (a1[i] > a1[i + 1]) {
                swapping_array(a1, i, i + 1);
            }
        } else {
            if (a1[i] < a1[i + 1]) {
                swapping_array(a1, i, i + 1);
            }
        }
    }
}

public static void swapping_array(int[] a1, int i, int k) {
    int temp = a1[i];
    a1[i] = a1[k];
    a1[k] = temp;
}

public static void swapping_array(int[] a1, int i, int k) {
    int temp = a1[i];
    a1[i] = a1[k];
    a1[k] = temp;
}

public static void main(String[] args) {
    int[] a1 = {7, 3, 15, 9, 21, 67, 2};

    System.out.println("The Original array is : " + Arrays.toString(a1));

    Zig_ZAG(a1);
    System.out.println("Updated Array in zigzag pattern: " + Arrays.toString(a1));
}
```

Output 05:

```
Output - LAB NO 4 (run)

run:
The Original array is : [7, 3, 15, 9, 21, 67, 2]
Updated Array in zigzag pattern: [3, 15, 7, 21, 9, 67, 2]
BUILD SUCCESSFUL (total time: 0 seconds)
|
```