# SIKSHA 'O' ANUSANDHAN
## DEEMED TO BE UNIVERSITY

**Admission Batch:** **Session:**

---

## Laboratory Record

## Computer Science Workshop 2 (CSE 3141)

***Submitted by***

Name: Saswat Mohanty

Registration No.: 1941012407

Branch: Computer Science and Engineering (CSE)

Semester: 4th Section: D


**Department of Computer Science & Engineering**

**Faculty of Engineering & Technology (ITER)**

**Jagamohan Nagar, Jagamara, Bhubaneswar, Odisha - 751030**

Scanned with CamScanner

# INDEX



Scanned with CamScanner

# Assignment on Array

Q1) write a program to implement the dutch national flag partition

Programme:-

```
public class DutchNationalFlag {
  public static void main (String[] args) {
     int [] value = {0,0,1,2,0,0,1,1,2}
     int [] arr = flag (value);
     print Array (arr);
  }
  public static int [] flag (int [] arr) {
     int [] a = new int [arr.length];
     for (int i=0; i<a.length; i++) {
          a[i] = arr[i];
      }
      int low=0, mid=0, high= a.length-1;
     while (mid<=high) {
          if (a[mid]=0) {
               int t = a[mid];
               a[mid] = a[low];
               a[low] = t;
                mid++;
                low++;
           }
          else if (a[mid]==2) {
                mid++;
          }
          else {
               int temp = a[mid];
               a[mid] = a[high];
               a[high] = temp
                high--;
```



Scanned with CamScanner

```
        }
      }
      return a;
    }
  public static void PrintArray(int[] a) {
    for(int i=0; i<a.length; i++) {
      System.out.println(a[i]+" ");
    }
  }
}
```

Output :-

```
0
0
0
0
2
2
1
1
1
```

Q2) write a program which takes as input an array of digits encoding a decimal number D and update the array to represent the number D+1.

Programmer :-

```
public class Increment {
  public static void main(String[] args) {
    int[] arr = {20, 25, 32, 12, 6, 10, 999, 150};
    int n = arr.length;
    System.out.println("Before increment");
    for(int i=0; i<n; i++) {
      System.out.println(arr[i]);
      System.out.print(" ");
    }
    for(int i=0; i<n; i++) {
```



Scanned with CamScanner

```
        arr[i] = arr[i] + 1;
    }
    System.out.println(" ");
    System.out.println("After increment");
    for(int i=0; i<n; i++){
        System.out.print(arr[i]);
        System.out.print(" ");
    }
  }
}
```

Output :-

{21, 26, 33, 13, 7, 11, 1000, 151}

Q3) write a program that takes two arrays representing integer, and return an integer representing their product.

Program :-

```
import java.util.*;
import java.util.stream.collectors;

public class ArbitraryPrecision {
  public static void main(String[] args) {
    List<Integer> n1 = Arrays.asList(1,9,3);
    List<Integer> n2 = Arrays.asList(-1,7,6);
  }
  static List<Long> MultiplyM2(List<Integer>num1, List<Integer>num2){
    long product = Integer.parseInt(num1.stream().map(Object::
                                    toString).collect(collectors.
                                    joining(" "))) *
                   Integer.parseInt(num2.stream().map(Object::
                                    toString).collect(collectors.
                                    joining(" ")));
    long productAbs = Math.abs(product);
```



Scanned with CamScanner

```
List <long> num3 = new ArrayList <long> ();
while (productAbs > 0) {
        num3.add (0, productAbs % 10);
        productAbs = productAbs / 10;
  }
  if (product < 0) {
    num3. set (0, -1 * num3.get (0));
  }
  system. out. println (num3);
  return num3;
}
```

Output :-

[-3, 3, 9, 6, 8]

Q4) write a program which takes as input a sorted array and updated it so that all duplicate have been removed and remaining elements shifted left to fill the emptied indices.

Program :-

```
import java.util. Arrays;
public class RemoveDuplicateFromArray {
  static int [] sortedArray (int [] nums) {
    for (int i = 0; i < nums. length; i++) {
      for (int j = 1; j < nums. length; j++) {
        if (nums [j-1] > nums [j]) {
          int temp = nums [j-1];
          nums [j-1] = nums [j];
          nums [j] = temp;
        }
      }
    }
    return nums;
  }
  static int newArray (int [] nums) {
    nums = sortedArray (nums);
```



Scanned with CamScanner

```
    int index = 1;
    for (int j=0; j<nums.length-1; j++) {
      if (nums[j] != nums[j+1]) {
        nums[index++] = nums[j+1];
      }
    }
    return index;
  }
  public static void main (String[] args) {
    int[] arr = {0,1,1,3,1,2,2,3,3,0};
    int length = newArray(arr);
    for (int i=0; i<length; i++) {
      System.out.print(arr[i] + " ");
    }
  }
}
```

Output :-

0 1 2 3

25) Write a program that takes an array denoting the daily stock price, and return the maximum profit that could be made by buying and then selling one share of that stock.

Program :-

```
public class BuyAndSell {
  public static void main (String[] args) {
    int a[] = {310, 315, 295, 260, 270, 230, 255, 250};
    int buy=0, sell=0, profit = -1, buyC = 0;
    for (int i=0; i<a.length; i++) {
      if (a[i] < a[buyC]) {
        buyC = i;
      }
      else if ((a[i] - a[buyC]) > profit) {
        profit = a[i] - a[buyC];
        sell = a[i];
    } }
```



Scanned with CamScanner

```
    System.out.println(profit + " " + buy + " " + sell);
  }
}
```

Output :-

30, 260, 290

Q6) Write a program that computes the maximum profit that can be made by buying and selling a share at most twice. The second buy must be made on another date after the first sale.

Program :-

```
public class BuyAndSellTwice {
  public class void main(String[] args) {
    double[] price = {10, 12, 13, 8, 12, 7, 11, 13, 15};
    System.out.println("Max profit: " + buyAndSellStockTwice(price));
  }
  public static double buyAndSellStockTwice(double[] stockPrices) {
    double buy1, buy2, profit1, profit2;
    buy1 = buy2 = Double.MAX_VALUE;
    profit1 = profit2 = 0;
    for (int i=0; i < stockPrices.length; i++) {
      buy1 = Math.min(buy1, stockPrices[i]);
      profit = Math.max(profit1, stockPrices[i] - buy1);
      System.out.println("Buy1: " + buy1 + " profit1: " + profit1);

      buy2 = Math.min(buy2, stockPrices[i] - profit1);
      profit2 = Math.max(profit2, stockPrices[i] - buy2);
      System.out.println("Buy2: " + buy2 + "profit2: " + profit2);
    }
    return profit2;
  }
}
```

output :-

Buy-2: 7.0 profit2 : 7.0
Buy-1: 7.0 profit1 : 7.0



Scanned with CamScanner

Buy-2: 7.0 profit2: 7.0
Buy-1: 7.0 profit1: 8.0
Buy-2: 7.0 profit2: 8.0

Max profit - 12.0

Q7) Write a program that takes an integer argument and returns all the primes between 1 and the integer.

Program:-

```
import java.util.ArrayList;
public class primeNumber {
 public static void main (String [] args) {
  int n=30;
  List < Boolean> list = new ArrayList <> ();
  for (int i=0; i<=n; i++) {
    list.add (true);
  }
   list.set (0, false);
   list.set (1, false);
   for (int j=2; j< list.size(); j++) { if (list.get(j) == true) {
     int i = j*2;
     while (i < list.size()) {
       list.set (i, false);
       i+=j;
     }
   }
 }
   for (int i=0; i<=n; i++) {
     list.add (true);
   }
   list.set (0, false);
   list.set (1, false);
   for (int j=2; j< list.size(); j++) {
     If (list.get(j) == true) {
      int i = j *2;
```



Scanned with CamScanner

```
        while (i < list.size()) {
          list.set(i, false);
          i += j;
        }
      }
    }
    System.out.println(list);
    System.out.println("Prime Numbers");
    for (int k=2; k<list.size(); k++) {
      if (list.get(k) == true) {
      }
    }
  }
}
```

Output :-

[2, 3, 5, 7]

Q8) Given an array A of n elements and a permutation P, apply P to A

Program :-

```
import java.util.*;
public class A2Q8 {
  public static void applyPermutation(List<character> A, List<Integer> P) {
    for (int i=0; i<A.size(); ++i) {
      int next = i;
      while (P.get(next) >= 0) {
        Collections.swap(A, i, P.get(next));
        int temp = P.get(next);
        P.set(next, P.get(next) - P.size());
        next = temp;
      }
    }
  }
}
```



Scanned with CamScanner

```
public static void main (String [] args){
  List <character> l = new ArrayList<> ();
  List <Integer> l2 = new ArrayList <>();
  Collections.addAll (l,'a','b','c','d');
  Collections.addAll (l2, 2, 0, 1, 3);
  applyPermutation (l, l2);
  System.out.println (l);
 }
}
```

Output:-

[b, c, a, d]

Q9) Write a program that takes as input a permutation, and returns the next permutation under dictionary ordering. If the permutations is the last permutation, return the empty array. For example, if the input is <1,0,3,2> your function should return <1,2,0,3>. If the input is <3,2,1,0>, return ().

Program:-

```
import java.util.*;
public class A209 {
 public static List <Integer> nextPermutation (List <Integer> l) {
  int k = l.size()-2;
  while (k>=0 && l.get(k)>= l.get(k+1)) {
    k--;
  }
  if (k==-1) {
    return Collections.emptyList();
  }
  for (int i = l.size()-1; i>k; --i){
    if (l.get(i) > l.get(k)) {
      Collections.swap (l,k,i);
      break;
    }
  }
```



Scanned with CamScanner

```
    Collections.reverse(l.subList(k+1, l.size()));
    return l;
  }
  public static void main(String[] args){
    List<Integer> l = new ArrayList<Integer>();
    Collections.addAll(1,65,66,67);
    nextPermutation(l);
    System.out.println(l);
  }
}
```

Output:-

[65, 67, 66]

Q10) Implement an algorithm that takes as input an array of distinct elements and a size, and returns a ~~sub~~ subset of the given size of the array elements. All subsets should be equally likely. Return the result in input array itself.

Program:-

```
import java.util.*;
public class A2Q10{
  public static void randomSample(int k, List<Integer> A){
    Random r = new Random();
    for(int i=0; i<k; i++){
      int index = i + r.nextInt(A.size()-i);
      Collections.swap(A, i, index);
    }
  }
  public static void display(){
    List<Integer> l = new ArrayList<Integer>();
    Collections.addAll(1,3,7,5,11);
    randomSample(3,l);
    int c=0;
    for(int i:l){
```



Scanned with CamScanner

```
    if (c == 3)
        break;
     System.out.print(i+" ");
     c++;
    }
 }
 public static void main (String[] args) {
   display();
  }
}
```

Output :-

7 3 5

Q11) Write a program that takes as input a positive integer n and a size $k < n$, and returns a size-k subset of $0, 1, 2, \ldots, n-1$. The subset should be represented as an array. All subsets should be equally likely and, in addition, all permutations of elements of the array should be equally likely. You may assume you have a function which takes as input a non-negative integer t and returns an integer in the set $0, 1, \ldots, t-1$ with uniform probability.

Program :-

```
import java.util.*;
public class A2Q11 {
  public static List<Integer> randomSubset(int k, int n) {
    Map<Integer, Integer> m = new HashMap<>();
    Random r = new Random();
    for (int i=0; i<k; i++) {
       int ran = i + r.nextInt(n-i);
       Integer p1 = m.get(ran);
       Integer p2 = m.get(i);
       if(p1 == null && p2 == null){
          m.put(ran, i);
          m.put(i, ran);
       }
```



Scanned with CamScanner

```
        else if (p1 == null && p2 != null) {
            m.put(ran, p2);
            m.put(i, ran);
        }
        else if (p1 != null && p2 == null) {
            m.put(i, p1);
            m.put(ran, i);
        }
        else {
            m.put(i, p1);
            m.put(ran, p2);
        }
    }
    List<Integer> res = new ArrayList<>(k);
    for(int i = 0; i < k; i++) {
        res.add(m.get(i));
    }
    return res;
}
public static void main(String[] args) {
    System.out.println(randomSubset(6, 14));
}
}
```

Output:-

[5, 6, 3, 12, 2, 1]

Q12) Check whether a 9X9 2D array representing a partially completed sudoku is valid. Specifically, check that no row, column, or 3X3 2D subarray contains duplicates. A 0-value in the 2D array indicates that entry is blank; every other entry is in [1,9].



Scanned with CamScanner

Program :-

```
import java.util.*;
public class A2Q12 {
  public static boolean isValidSudoku(List<List<Integer>> partialAssignment) {
    for (int i=0; i < partialAssignment.size(); ++i) {
      if(hasDuplicate(partialAssignment, i, i+1, 0, partialAssignment.size())) {
        return false;
      }
    }
    for (int j=0; j<partialAssignment.size(); ++j) {
      if (hasDuplicate(partialAssignment, 0, partialAssignment.size(),
                       j, j+)) {
        return false;
      }
    }
    int regionSize = (int) Math.sqrt(partialAssignment.size());
    for (int I=0; I< regionSize; ++I) {
      for(int J=0; J< regionSize; ++J) {
        if (hasDuplicate(partialAssignment, regionSize * I,
                         regionSize * (I+1), regionSize * J,
                         regionSize * (J+1))) {
          return false;
        }
      }
    }
    return true;
  }
  private static boolean hasDuplicate(List<List<Integer>> partialA-
      -ssignment, int startRow, int endRow, int startCol,
      int endCol) {
```



Scanned with CamScanner

```
List <Boolean> isPresent = new ArrayList <> (Collections.nCopies(
                                partialAssignment.size()+1, false));
for (int i= startRow; i< endRow; ++i) {
   for(int j= startCol; j<endCol; ++j) {
      if (partialAssignment.get(i).get(j) != 0 &&
         isPresent.get (partialAssignment.get(i).get(j))) {
            return true;
      }
      isPresent.set (partialAssignment.get(i).get(j), true);
   }
}
return false;
}
public static void main (String[] args) {
List <List <Integer>> l = new ArrayList <List <Integer>>();
l.add (new ArrayList <Integer> (Arrays.asList (5, 3,0,0, 7,0, 0,
                                0,0)));
l.add (new ArrayList <Integer> (Arrays.asList (6,0,0,1,9,5,0,
                                0,0)));
l.add (new ArrayList <Integer> (Arrays.asList (0,9,8,0,0,0, 0,
                                6,0)));
l.add (new ArrayList <Integer> (Arrays.asList (8,0,0,0,6,0,
                                0,0,3)));
l.add (new ArrayList <Integer> (Arrays.asList (4,0,0,8,0,
                                3,0,0,1)));
l.add (new ArrayList <Integer> (Arrays.asList (7,0,0,0,2,0,
                                0, 0, 6)));
l.add (new ArrayList <Integer> (Arrays.asList (0,6,0,0,0,0,8,8,
                                0)));
l.add (new ArrayList <Integer> (Arrays.asList (0,0,0,4,1,9,0,
                                0,5)));
```



Scanned with CamScanner

```
l.add(new ArrayList<Integer>(Arrays.asList(0,0,0,0,8,0,
                                           0,7,9)));
System.out.println("Partial sudoku configurations:");
for(int i=0; i<l.size(); i++){
    for(int j=0; j<l.get(i).size(); j++){
        System.out.print(l.get(i).get(j)+" ");
    }
}
boolean result = isValidSudoku(l);
if(result){
    System.out.println("The board is valid.");
}
else{
    System.out.println("The board is invalid.");
}
}
}
```

Output:-

Partial sudoku configurations

```
5 3 0 0 7 0 0 0 0
6 0 0 1 9 5 0 0 0
0 9 8 0 0 0 0 6 0
8 0 0 0 6 0 0 0 3
4 0 0 8 0 3 0 0 1
7 0 0 0 2 0 0 0 6
0 6 0 0 0 0 2 8 0
0 0 0 4 1 9 0 0 5
0 0 0 0 8 0 0 7 9
```

The board is valid.



Scanned with CamScanner

Q13) Write a program which takes an nxn 2D array and returns the spiral ordering of the array.

Program :-

```
import java.util.*;
public class A2Q13 {
  public static List<Integer> matrixInSpiralOrder(List<List<Integer>>
                                squareMatrix) {
    List<Integer> spiralOrdering = new ArrayList<>();
    for (int offset = 0; offset < Math.ceil(0.5*squareMatrix.size());
                                ++offset) {
      matrixLayerInClockwise(squareMatrix, offset, spiralOrdering);
    }
    return spiralOrdering;
  }
  private static void matrixLayerInClockwise(List<List<Integer>>
          squareMatrix, int offset, List<Integer> spiralOrdering) {
    if (offset == squareMatrix.size() - offset - 1) {
      spiralOrdering.add(squareMatrix.get(offset).get(offset));
      return;
    }
    for (int j=offset; j<squareMatrix.size()-offset-1; ++j) {
      spiralOrdering.add(squareMatrix.get(offset).get(j));
    }
    for (int i=offset; i<squareMatrix.size()-offset-1; ++i) {
      spiralOrdering.add(squareMatrix.get(i).get(squareMatrix
                        .size()-offset-1));
    }
    for (int j = squareMatrix.size()-offset-1; j>offset; --j) {
      spiralordering.add(squareMatrix.get(squareMatrix.size()
                        -offset-1).get(j));
    }
  }
```



Scanned with CamScanner

```
    for (int i = squareMatrix.size() - offset - 1; i > offset; --i) {
        spiralOrdering.add(squareMatrix.get(i).get(offset));
    }
}

public static void main (String[] args) {
    List<List<Integer>> squareMatrix = new ArrayList<List<
                                        Integer>>();
    squareMatrix.add(new ArrayList<Integer>(Arrays.asList(1,
                                        2, 3, 4)));
    squareMatrix.add(new ArrayList<Integer>(Arrays.asList(5, 6,
                                        7, 8)));
    squareMatrix.add(new ArrayList<Integer>(Arrays.asList(9,
                                        10, 11, 12)));
    squareMatrix.add(new ArrayList<Integer>(Arrays.asList(13,
                                        14, 15, 16)));
    System.out.println("The Input Matrix is: ");
    for(int i = 0; i < squareMatrix.size(); i++) {
        for(int j = 0; j < squareMatrix.get(i).size(); j++) {
            System.out.println(squareMatrix.get(i).get(j)+" ");
        }
        System.out.println();
    }
    List<Integer> spiralOrdering = new ArrayList<>();
    spiralordering = matrixInSpiralOrder(squareMatrix);
    System.out.println("The spiral ordering of elements in the
                                        given matrix: ");
    for(int num: spiralordering) {
        System.out.print(num + " ");
    }}}
```



Scanned with CamScanner

Output :-

```
The Input Matrix :
1   2   3   4
5   6   7   8
9   10  11  12
13  14  15  16
The Spiral ordering of elements in the given matrix
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
```

Q14) Write a function that takes as input an $n \times n$ 2D array and rotates the array by 90 degrees clockwise.

Program :-

```
import java.util.*;
public class A2Q14 {
    public static void rotate(int[][] matrix, int len) {
        int[][] newmatrix = new int[len][len];
        for(int i=0; i<len; i++) {
            for(int j=0; j<len; j++) {
                int x = matrix[i][j];
                newmatrix[j][len-1-i] = x;
            }
        }
        for (int i=0; i<len; i++) {
            for(int j=0; j<len; j++) {
                matrix[i][j] = newmatrix[i][j]
            }
        }
    }
    public static void print(int[][] matrix, int n) {
        System.out.println("\n2-D Array: ");
```



Scanned with CamScanner

```
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            System.out.print(matrix[i][j]+" ");
        System.out.println();
    }
}
public static void main (String[] args){
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the value of N: ");
    int n = sc.nextInt();
    int matrix[][] = new int[n][n];
    int c = 1;
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            matrix[i][j] = c;
            c = c+1;
        }
    }
    print(matrix, n);
    int len = matrix.length;
    rotate(matrix, len);
    print(matrix, n);
  }
}
```

Output :-

```
Enter the value of N
5
2-D array:
1  2  3  4  5
```



Scanned with CamScanner

| | | | | |
|---|---|---|---|---|
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 |

2-D Array:

| | | | | |
|---|---|---|---|---|
| 21 | 16 | 11 | 6 | 1 |
| 22 | 17 | 12 | 7 | 2 |
| 23 | 18 | 13 | 8 | 3 |
| 24 | 19 | 14 | 9 | 4 |
| 25 | 20 | 15 | 10 | 5 |

Q15) Write a program which takes as input a non-negative integer n and returns the first n rows of Pascal's triangle.

Program :-

```
import java.util.*;
public class A2Q15 {
    public static List<List<Integer>> generate(int numRows) {
        int count = 2
        List<List<Integer>> result = new ArrayList<>();
        result.add(Arrays.asList(1));
        List<Integer> newTempList, currentList;
        while (count <= numRows) {
            currentList = result.get(result.size() - 1);
            newTempList = new ArrayList<>();
            newTempList.add(1);
            for(int i=0; i < currentList.size() - 1; i++) {
                newTempList.add(currentList.get(i) +
                                currentList.get(i+1));
            }
            newTempList.add(1)
            result.add(newTempList);
            count ++
```



Scanned with CamScanner

```
        }
        return result;
    }
    public static void main (String [] args) {
        List <List <Integer>> res = generate(3);
        String result = "";
        for(int i=0; i<res.size(); i++) {
            for(int j=0; j<res.get(i).size(); j++) {
                result += res.get(i).get(j)+" ";
            }
            result += "\n";
        }
        System.out.println(result);
    }
}
```

Output :-

```
1
1  1
1  2  1
```



Scanned with CamScanner