**EXERCISE -** 10

**N Queens, OBST & KnapSack(Dynamic)**

**Aim:** Write a java program to implement N Queens problem.

**File name:** NQueens.java

**Program:**

**import java.util.Arrays;**

**public class NQueens {**

**public static void main(String[] args) {**

**int N = 4;**

**int[] Q = new int[N];**

**Dynamic(Q, N, 0);**

**// System.out.println(Arrays.toString(Q));**

**}**

**private static void Dynamic(int[] q, int n, int row) {**

**if (row == n) {**

**System.out.println("Columns"+Arrays.toString(q));**

**return;**

**}**

**for (int col = 0; col < n; col++) {**

**q[row] = col;**

**if (isSafe(q, row)) { Dynamic(q, n, row + 1);}**

**}**

**}**

**private static boolean isSafe(int[] q, int row) {**

**for (int i = 0; i < row; i++) {**

**if (q[i] == q[row] || Math.abs(q[i] - q[row]) == Math.abs(i - row)) {**

**return false;**

**}**

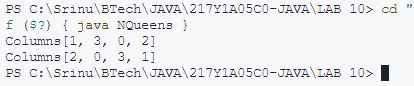
**}**

**return true;**

**}**

**}**

**Output:**

****

**Aim:** Write a java program to implement Optimal Binary search tree problem.

**File name:** \_filename\_.java

**Program:**

**public class OptimalBST{**

**public static void main(String[] args) {**

**int keys[] = {3,3,1,1};**

**int freq[] = {2,3,1,1,1};**

**int n = keys.length;**

**System.out.println("Cost of Optimal BST is"+optimalSearchTree(keys, freq, n));**

**}**

**static int optCost(int freq[], int i, int j){**

**if (j < i)**

**return 0;**

**if (j == i)**

**return freq[i];**

**int fsum = sum(freq, i, j);**

**int min = Integer.MAX\_VALUE;**

**for (int r = i; r <= j; ++r){**

**int cost = optCost(freq, i, r-1) + optCost(freq, r+1, j);**

**if (cost < min)**

**min = cost;**

**}**

**return min + fsum;**

**}**

**static int optimalSearchTree(int keys[], int freq[], int n){**

**return optCost(freq, 0, n-1);}**

**static int sum(int freq[], int i, int j){**

**int s = 0;**

**for (int k = i; k <=j; k++)**

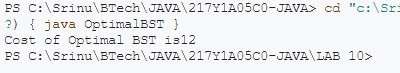
**s += freq[k];**

**return s;**

**}**

**}**

**Output:**

****

**Aim:** Write a java program to implement KnapSack problem.

**File name:** KnapsackDY.java

**Program:**

**// import java.lang.\*;**

**class KnapsackDY{**

**public static void main(String[] args)**

**{**

**int profit[] = { 60, 100, 120 };**

**int weight[] = { 10, 20, 30 };**

**int W = 50;**

**int n = profit.length;**

**System.out.print(KnapSack(W, weight, profit, n));**

**}**

**static int KnapSack(int W,int wt[],int val[],int n){**

**int[] dp = new int[W + 1];**

**for (int i = 1; i < n + 1; i++) {**

**for (int w = W; w >= 0; w--) {**

**if (wt[i - 1] <= w)**

**dp[w] = Math.max(dp[w], dp[w - wt[i - 1]]+ val[i - 1]);**

**}**

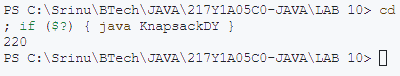
**}**

**return dp[W];**

**}**

**}**

**Output:**

****