Practical 1

Input:

```
import java.util.Scanner;


public class FibonacciWithStepCount {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the number of Fibonacci numbers to calculate: ");
    int n = scanner.nextInt();


    long[] fibonacciSeries = new long[n];
    long stepCount = 0;


    if (n >= 1) {
      fibonacciSeries[0] = 0;
      stepCount++;
    }
    if (n >= 2) {
      fibonacciSeries[1] = 1;
      stepCount++;
    }


    for (int i = 2; i < n; i++) {
      fibonacciSeries[i] = fibonacciSeries[i - 1] + fibonacciSeries[i - 2];
      stepCount += 3; // Each iteration includes 3 steps: addition, access previous two values, and
assignment.
    }
```

```
        System.out.println("Fibonacci Series:");

        for (int i = 0; i < n; i++) {

            System.out.print(fibonacciSeries[i] + " ");

        }


        System.out.println("\nTotal steps taken to calculate the Fibonacci series: " + stepCount);

    }

}
```

Output:

```
Enter the number of Fibonacci numbers to calculate: 40

Fibonacci Series:

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368
75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887 9227465
14930352 24157817 39088169 63245986

Total steps taken to calculate the Fibonacci series: 116
```

Practical 2

Input:

```
import java.util.*;
public class job
{
  public static void main(String args[])
  {
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter the number of Jobs");
    int n=sc.nextInt();
    String a[]=new String[n];
    int b[]=new int[n];
    int c[]=new int[n];
    for(int i=0;i<n;i++)
    {
      System.out.println("Enter the Jobs");
      a[i]=sc.next();
      System.out.println("Enter the Profit");
      b[i]=sc.nextInt();
      System.out.println("Enter the DeadLine");
      c[i]=sc.nextInt();
    }
    System.out.println("--Arranged Order--");
    System.out.print("Jobs:    ");
    for(int i=0;i<n;i++)
    {
      System.out.print(a[i]+" ");
    }
    System.out.println();
    System.out.print("Profit:  ");
    for(int i=0;i<n;i++)
    {
      System.out.print(b[i]+" ");
    }
    System.out.println();
    System.out.print("DeadLine:");
    for(int i=0;i<n;i++)
    {
      System.out.print(c[i]+" ");
    }
    for(int i=0;i<n-1;i++)
    {
      for(int j=i+1;j<n;j++)
      {
          if(b[i]<b[j])
          {
              int temp=b[i];
              b[i]=b[j];
               b[j]=temp;

              temp=c[i];
              c[i]=c[j];
               c[j]=temp;

              String temp1=a[i];
              a[i]=a[j];
               a[j]=temp1;
          }
      }
    }
    System.out.println();
    System.out.println("--Sorted Order--");
    System.out.print("Jobs:    ");
    for(int i=0;i<n;i++)
    {
      System.out.print(a[i]+" ");
    }
    System.out.println();
    System.out.print("Profit:  ");
    for(int i=0;i<n;i++)
    {
      System.out.print(b[i]+" ");
    }
    System.out.println();
    System.out.print("DeadLine:");
    for(int i=0;i<n;i++)
    {
      System.out.print(c[i]+" ");
    }
    System.out.println();
    int max=c[0];
    for(int i=0;i<n;i++)
    {
      if(c[i]>max)
      {
        max=c[i];
      }
    }
    String x[]=new String[max];
    int xx[]=new int[max];
    int profit=0;
    for(int i=0;i<n;i++)
    {
      int pp=c[i];
      pp=pp-1;
      if(x[pp]==null )
      {
        x[pp]=a[i];
        profit+=b[i];
      }
      else
      {
        while(pp!=-1)
        {
          if(x[pp]==null)
          {
            x[pp]=a[i];
            profit+=b[i];
            break;
          }
            pp=pp-1;
        }
      }
    }
    for(int i=0;i<max;i++)
    {
        System.out.print("-->"+x[i]);
    }
    System.out.println();
```

```
    System.out.print("Profit Earned"+profit);
 }
}
```

Output:

```
Enter the number of Jobs

7

Enter the Jobs

J1

Enter the Profit

35

Enter the DeadLine

3

Enter the Jobs

J2

Enter the Profit

30

Enter the DeadLine

4

Enter the Jobs

J3

Enter the Profit

25

Enter the DeadLine

4

Enter the Jobs

J4

Enter the Profit

20
```

```
Enter the DeadLine

2

Enter the Jobs

J5

Enter the Profit

15

Enter the DeadLine

3

Enter the Jobs

J6

Enter the Profit

12

Enter the DeadLine

1

Enter the Jobs

J7

Enter the Profit

5

Enter the DeadLine

2

--Arranged Order--

Jobs:    J1 J2 J3 J4 J5 J6 J7

Profit:  35 30 25 20 15 12 5

DeadLine:3 4 4 2 3 1 2

--Sorted Order--

Jobs:    J1 J2 J3 J4 J5 J6 J7

Profit:  35 30 25 20 15 12 5
```

```
DeadLine:3 4 4 2 3 1 2

-->J4-->J3-->J1-->J2

Profit Earned110
```

Practical 3

Input:

```
import java.io.IOException;

import java.util.Scanner;


class Fractional_Knapsack

{

   public static void main(String args[]) throws IOException

   {

      int i,j=0,max_qty,m,n;

      float sum=0,max;

      Scanner sc = new Scanner(System.in);

      int array[][]=new int[2][20];

      System.out.println("Enter no of items");

      n=sc.nextInt();


      System.out.println("Enter the weights of each items");

      for(i=0;i<n;i++)

         array[0][i]=sc.nextInt();


      System.out.println("Enter the values of each items");

      for(i=0;i<n;i++)

         array[1][i]=sc.nextInt();


      System.out.println("Enter maximum volume of knapsack :");

      max_qty=sc.nextInt();
```

```
m=max_qty;

while(m>=0)

{

   max=0;

   for(i=0;i<n;i++)

   {

      if(((float)array[1][i])/((float)array[0][i])>max)

      {

         max=((float)array[1][i])/((float)array[0][i]);

         j=i;

      }

   }

   if(array[0][j]>m)

   {

      System.out.println("Quantity of item number: " +  (j+1) + " added is " +m);

      sum+=m*max;

      m=-1;

   }

   else

   {

      System.out.println("Quantity of item number: " + (j+1) + " added is " + array[0][j]);

      m-=array[0][j];

      sum+=(float)array[1][j];

      array[1][j]=0;

   }

}

System.out.println("The total profit is " + sum);
```

```
        sc.close();
    }
}
```

Output:

```
Enter no of items
5
Enter the weights of each items
10 20 30 40 50
Enter the values of each items
5 4 3 2 1
Enter maximum volume of knapsack :
80
Quantity of item number: 1 added is 10
Quantity of item number: 2 added is 20
Quantity of item number: 3 added is 30
Quantity of item number: 4 added is 20
The total profit is 13.0
```

Practical 4

Input:

```
import java.util.Scanner;


public class Zero_One_Knapsack

{

    public void solve(int[] wt, int[] val, int W, int N)

    {

        int NEGATIVE_INFINITY = Integer.MIN_VALUE;

        int[][] m = new int[N + 1][W + 1];

        int[][] sol = new int[N + 1][W + 1];

        for (int i = 1; i <= N; i++)

        {

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

            {

                int m1 = m[i - 1][j];

                int m2 = NEGATIVE_INFINITY;

                if (j >= wt[i])

                    m2 = m[i - 1][j - wt[i]] + val[i];

                m[i][j] = Math.max(m1, m2);

                sol[i][j] = m2 > m1 ? 1 : 0;

            }

        }

        int[] selected = new int[N + 1];

        for (int n = N, w = W; n > 0; n--)

        {

            if (sol[n][w] != 0)
            {
                selected[n] = 1;
                w = w - wt[n];
            }
            else
                selected[n] = 0;
        }
        System.out.print("\nItems with weight ");
        for (int i = 1; i < N + 1; i++)
            if (selected[i] == 1)
                System.out.print(val[i] +" ");
        System.out.println("are selected by knapsack algorithm.");
    }
    public static void main (String[] args)
    {
        Scanner scan = new Scanner(System.in);
        Zero_One_Knapsack ks = new Zero_One_Knapsack();

        System.out.println("Enter number of elements ");
        int n = scan.nextInt();

        int[] wt = new int[n + 1];
        int[] val = new int[n + 1];

        System.out.println("Enter weight for "+ n +" elements");
        for (int i = 1; i <= n; i++)
            wt[i] = scan.nextInt();
        System.out.println("Enter value for "+ n +" elements");
        for (int i = 1; i <= n; i++)
            val[i] = scan.nextInt();

        System.out.println("Enter knapsack weight ");
        int W = scan.nextInt();

        ks.solve(wt, val, W, n);
        scan.close();
    }
}
```

Outpt:

```
Enter number of elements
5
Enter weight for 5 elements
01 56 42 78 12
Enter value for 5 elements
50 30 20 10 50
Enter knapsack weight
150

Items with weight 50 30 20 50 are selected by knapsack algorithm.
```

Practical 5

Input:

```
package com.JournalDev;

public class Main {

    static final int N = 4;


   // print the final solution matrix

    static void printSolution(int board[][])

    {

        for (int i = 0; i < N; i++) {

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

                System.out.print(" " + board[i][j]

                        + " ");

            System.out.println();

        }

    }


    // function to check whether the position is safe or not

    static boolean isSafe(int board[][], int row, int col)

    {

        int i, j;

        for (i = 0; i < col; i++)

            if (board[row][i] == 1)

                return false;



        for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
            if (board[i][j] == 1)
                return false;

        for (i = row, j = col; j >= 0 && i < N; i++, j--)
            if (board[i][j] == 1)
                return false;

        return true;
    }

    // The function that solves the problem using backtracking
    public static boolean solveNQueen(int board[][], int col)
    {
        if (col >= N)
            return true;

        for (int i = 0; i < N; i++) {
            //if it is safe to place the queen at position i,col -> place it
            if (isSafe(board, i, col)) {
                board[i][col] = 1;

                if (solveNQueen(board, col + 1))
                    return true;

                //backtrack if the above condition is false
                board[i][col] = 0;
            }
        }
        return false;
    }

    public static void main(String args[])
    {
        int board[][] = { { 0, 0, 0, 0 },
                { 0, 0, 0, 0 },
                { 0, 0, 0, 0 },
                { 0, 0, 0, 0 } };

        if (!solveNQueen(board, 0)) {
            System.out.print("Solution does not exist");
            return;
        }

        printSolution(board);

    }
}
```

Output :

```
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
```