Computer Science
Programming Assignment

A compilation of programs and algorithms encompassing
a variety of topics covered through the year

Done by

NAMIT NATHWANI

ISC - Grade 12
S. N. Kansagra School
2016–2017

Acknowledgement

This project would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this assignment.

I express utmost gratitude to my Computer Science teachers Ms. Nishat Khimani, Ms. Manaali Dubal, and Mr. Siddharth Naidu whose inputs and encouragement have been my inspiration as I hurdled over all the obstacles in the completion of this project. I would also like to thank them for providing valuable inputs and feedback to aid us in our project.

Furthermore, I thank all the members of my family who always had a kind concern and consideration regarding all my project and academic requirements and were helpful all through the project. I specially thank my father whose inputs regarding the details of this project were very valuable and helpful given his eye for detail. I also thank my mother, who helped sand off the rough edges of the project, giving feedback and suggestions regarding the aesthetic part of the project.

Contents

1 Array Fill

Take input of order of 2D matrix. Take 3 characters (c1, c2, c3) as user input and fill the diagonals with c3, the parts created on the top and bottom by the diagonals with c1 and the parts to the left and right with c2.
Example:

Input:
Order = 7
Characters: @, #, $

Output:

```
$ @ @ @ @ @ $
# $ @ @ @ $ #
# # $ @ $ # #
# # # $ # # #
# # $ @ $ # #
# $ @ @ @ $ #
$ @ @ @ @ @ $
```

1.1 Algorithm

1) Declare variables 'n' (For storing order of matrix), 'c1', 'c2', and 'c3' for storing the three characters and take appropriate user inputs.

2) Declare a character array M[ ][ ] of size n*n

3) Run loop from i = 0 and j = 0 till i < n, incrementing i and j at each iteration and repeat step A) each time

A) Assign c3 to M[i][j]

4) Run loop from i = n - 1 and j = 0 till i $\geq$ 0, incrementing j and decrementing i at each iteration and repeat step A) each time

A) Assign c3 to M[i][j]

5) Declare 'limit' to store limit of loops

6) if n % 2 = 0, go to step A), else go to step B)

A) limit = n/2 - 1

B) limit = n/2

7) Run loop from i = 0 and j = 0 till i < limit, incrementing i and j at each iteration and repeat step A) each time

A) Run loop from k = i + 1, till k < (n - i - 1), incrementing k at each iteration and repeat step i) each time

i) Assign c1 to M[i][k]

8) Run loop from i = n - 1 and j = 0 till j < limit, incrementing j and decrementing i at each iteration and repeat step A) each time

A) Run loop from k = j + 1, till k < (n - j - 1), incrementing k at each iteration and repeat step i) each time

i) Assign c1 to M[i][k]

9) Run loop from i = 0 and j = 0 till i < limit, incrementing i and j at each iteration and repeat step A) each time

A) Run loop from k = i + 1, till k < (n - i - 1), incrementing k at each iteration and repeat step i) each time

i) Assign c2 to M[k][i]

10) Run loop from i = n - 1 and j = 0 till j < limit, incrementing j and decrementing i at each iteration and repeat step A) each time

A) Run loop from k = j + 1, till k < (n - j - 1), incrementing k at each iteration and repeat step i) each time

i) Assign c1 to M[k][i]

11) Run loop from i = 0 till i < n, incrementing i at each iteration and repeat steps A) and B) each time

A) Run loop from j = 0 till j < n, incrementing j at each iteration and repeat step i) each time

i) Print M[i][j]

B) Print empty line

1.2 Code

```
import java.util.Scanner;
public class Arr_fill
{
   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      int n, i, j, k;
      char c1, c2, c3; //To store characters
      int limit; //To store loop limit (half of order)


      System.out.println("Enter order");
      n = s.nextInt(); //Input for order

      System.out.println("Enter 3 characters");
      c1 = s.next().charAt(0);
      c2 = s.next().charAt(0);
      c3 = s.next().charAt(0);
      char M[][] = new char[n][n]; //Array to fill

      for(i = 0, j = 0; i < n; i++, j++) //Fills left diagonal
         M[i][j] = c3;

      for(i = (n - 1), j = 0; i >= 0; i--, j++) //Fills Right Diagonal
         M[i][j] = c3;

      if(n % 2 == 0)
         limit = (n / 2) - 1;
      else
         limit = n / 2;

      for(i = 0, j = 0; i < limit; i++, j++) //To fill top part
         for(k = i + 1; k < n - 1 - i; k++)
            M[i][k] = c1;
```

```
      for(i = n - 1, j = 0; j < limit; i--, j++) //To fill bottom part
         for(k = j + 1; k < n - j - 1; k++)
            M[i][k] = c1;

      for(i = 0, j = 0; i < limit; i++, j++) //To fill left part
         for(k = i + 1; k < n - 1 - i; k++)
            M[k][i] = c2;

      for(i = n - 1, j = 0; j < limit; i--, j++) //To fill right part
         for(k = j + 1; k < n - j - 1; k++)
            M[k][i] = c2;

      for(i = 0; i < n; i++) //To Print Array
      {
         for(j = 0; j < n; j++)
            System.out.print(M[i][j] + " ");
         System.out.println();
      }
   }
}
```

2 Character Counter

Write a program to count the number of instances of each character in a String

Input:
Java J2EE Java JSP J2EE

Output:

```
J   :  5
a   :  4
v   :  2
    :  4
2   :  2
E   :  4
S   :  1
P   :  1
```

2.1 Algorithm

1) Declare a Scanner object to accept user input

2) Take user input of a String(inp)

3) Declare a char array (chars[ ]) and an int array (freq[ ]) of the same length as the input

4) Run a loop from i = 0 to length of inp - 1, incrementing i at each iteration

   A) Run a loop from j = 0 to length of chars[ ] - 1, incrementing j at each iteration

      i) if chars[j] is null,

         a) Set chars[j] to the current character in String

         b) Set frequency of the character to 1

         c) Break the loop

      ii) if chars[j] is present in the array

         a) Increase frequency of the character

         b) Break the loop

5) Print the characters and their frequencies

2.2 Code

```
import java.util.Scanner;

public class CharCount
{
   public static void main(String args[])
    {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter String");
      String inp = s.nextLine();      //Stores input String

      char[] chars = new char[inp.length()]; //char array of length = length of input to store
           each character
```

```
    int[] freq = new int[inp.length()];  //int array of length = length of input to store
        frequency of each character

    for(int i = 0; i < inp.length(); i++) //iterates through characters in string
    {
       for(int j = 0; j < chars.length; j++) //iterates through character array (chars)
       {
          if(chars[j] == Character.MIN_VALUE) //if chars[j] is null
          {
             chars[j] = inp.charAt(i);  //sets chars[j] to current character
             freq[j] = 1;            //increases frequency of that character to 1
             break;
          }
          if(chars[j] == inp.charAt(i))  //if character present in array
          {
             freq[j]++;              //increases frequency of character by 1
             break;
          }
       }
    }

    for(int i = 0; i < chars.length; i++)  //Prints characters and their frequency
    {
       if(chars[i] != Character.MIN_VALUE)
          System.out.println(chars[i] + " : " + freq[i]);
    }

    s.close();
   }
}
```

3 Decimal, Binary, Octal, Hexadecimal Convertor

Write a program to convert Decimal to Binary, Octal and Hexadecimal numbering systems.
Example:

Input:
13

Output:
Number in Binary: 1101
Number in Octal: 0o15
Number in Hexadecimal: 0xD

3.1 Algorithm

1) Declare a static int array rems[ ] (from 1 to 9 and A to F) to store remainders for use in conversion

2) Take user input for number to convert

3) Call all three functions and print the returned values

String DecToBin(int n):

1) If n = 0, return 0

2) Declare a String (Binary), and initialize it to an empty String

3) Run a loop from rem = n % 2, till n > 0, setting rem to n % 2 at each iteration

   A) Append rems[rem] to the front of Binary

   B) Set n = n / 2

4) Return Binary

String DecToOct(int n):

1) If n = 0, return 0

2) Declare a String (Octal), and initialize it to an empty String

3) Run a loop from rem = n % 8, till n > 0, setting rem to n % 8 at each iteration

   A) Append rems[rem] to the front of Octal

   B) Set n = n / 8

4) Return "0o" + Octal

String DecToHex(int n):

1) If n = 0, return 0

2) Declare a String (Hex), and initialize it to an empty String

3) Run a loop from rem = n % 16, till n > 0, setting rem to n % 16 at each iteration

   A) Append rems[rem] to the front of Hex

   B) Set n = n / 16

4) Return “0x” + Hex

3.2 Code

```
import java.util.Scanner;

public class DecToBin_Oct_Hex
{
   static char[] rems = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D',
       'E', 'F'}; //Array

   public static void main(String args[])
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter Number to Convert");
      int n = s.nextInt();

      System.out.println("Number in Binary: " + Q11_DecToBin_Oct_Hex.DecToBin(n)); //Prints
          Binary equivalent by calling Function
      System.out.println("Number in Octal: " + Q11_DecToBin_Oct_Hex.DecToOct(n));  //Prints
          Octal equivalent by calling Function
      System.out.println("Number in Hexadecimal: " + Q11_DecToBin_Oct_Hex.DecToHex(n));
          //Prints Hexadecimal equivalent by calling Function

      s.close();
   }

   public static String DecToBin(int n) //Function to return Binary equivalent of a decimal
       number
   {
      if(n == 0)
        return "0";

      String Binary = ""; //String to store Binary equivalent
      for(int rem = n % 2; n > 0; rem = n % 2) //Loop to append remainder to string, creating
          Binary number
      {
         Binary = rems[rem] + Binary;
         n = n / 2;
      }
      return Binary; //returns Binary equivalent
   }

   public static String DecToOct(int n) //Function to return Octal equivalent of a decimal
       number
   {
      if(n == 0)
        return "0";

      String Octal = ""; //String to store Octal equivalent
      for(int rem = n % 8; n > 0; rem = n % 8) //Loop to append remainder to string, creating
          Octal number
      {
         Octal = rems[rem] + Octal;
         n = n / 8;
      }
      return "0o" + Octal; //returns Octal equivalent
   }
```

```
  public static String DecToHex(int n) //Function to return Hexadecimal equivalent of a
      decimal number
  {
     if(n == 0)
        return "0";

     String Hex = ""; //String to store Hexadecimal equivalent
     for(int rem = n % 16; n > 0; rem = n % 16) //Loop to append remainder to string, creating
         Hexadecimal number
     {
        Hex = rems[rem] + Hex;
        n = n / 16;
     }
     return "0x" + Hex; //returns Hexadecimal equivalent
  }
}
```

4 Sorting Boundary Elements of a 2-D Array

Sort the elements of the outer rowns and columns in ascending order and calculate the sum of the boundary elements. Display the Rearranged Matrix and Boundary Element Matrix.

Input:
N = 3, M = 3
1 5 7
8 9 2
6 4 3

Output:
Rearranged Matrix:
1 2 3
8 9 4
7 6 5
Only Boundary Elements:
1 2 3
8 4
7 6 5
Sum Of Boundary Elements = 36

4.1 Algorithm

1) Take user input for dimensions of the array

2) Create a 2-D int array (a) with the given dimensions

3) Take user input for the data

4) Print the array

5) Declare a new int array (bounds) of size 2*(n + m - 2)

6) Run a loop from i = 0, j = 0 to i ¡ n, incrementing i and j

A) Set bounds[j] to a[0][i] (Elements of Row 1)

7) Run a loop from i = 1 to m - 2, incrementing i and j

A) Set bounds[j] to a[i][n - 1] (Elements of column (n - 1), from row 1 to row (m - 2))

8) Run a loop from i = n - 1 to 0, decrementing i and incrementing j

A) Set bounds[j] to a[m - 1][i] (Elements of row (m - 1), from columns (n - 2) to 1

9) Run a loop from i = m - 2 to 1, decrementing i and incrementing j

A) Set bounds[j] to a[i][0] (Elements of column 0, from rows (m - 1) to 1

10) Sort bounds in ascending order

11) Run a loop from i = 0, j = 0 to i ¡ n, incrementing i and j

A) Set a[0][i] to bounds[j] (Elements of row 0)

12) Run a loop from i = 1 to m - 2, incrementing i and j

A) Set a[i][n - 1] to bounds[j] (Elements of column (n - 1), rows 1 to (m-2))

13) Run a loop from i = n - 1 to 0, decrementing i and incrementing j

A) Set a[m - 1][i] to bounds[j] (Elements of row (m - 1), columns (n-2) to 1)

14) Run a loop from i = m - 2 to 1, incrementing j and decrementing i

A) Set a[i][0] to bounds[j] (Elements of column 0)

15) Print new Array

16) Run a loop through bounds[], adding each element to a new int (Sum), and display sum

4.2 Code

```
import java.util.Scanner;
public class BoundaryElementSort_Sum_Display
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter dimensions of the Array: row and column");
      int m = s.nextInt(), n = s.nextInt(), a[][] = new int[m][n], i, j, sum = 0; //Input of
          Array dimensions and declaration of array

      System.out.println("Enter data row-wise:");
      for(i = 0; i < m; i++)
         for(j = 0; j < n; j++)
            a[i][j] = s.nextInt(); //To Store user inputs

      System.out.println("The Input Array: ");
      for(i = 0; i < m; i++)
         for(j = 0; j < n; j++)
            System.out.println(a[i][j] + "\t"); //Displaying User Input array

      int bounds[] = new int[2*(n + m -2)];

      for(i = 0, j = 0; i < n; i++, j++)  //Storing elements from row 0
         bounds[j] = a[0][i];
      for(i = 1; i < m - 1; i++, j++)    //Storing elements of column (n - 1), from row 1 to
          row (m-2)
         bounds[j] = a[i][n - 1];
      for(i = n - 1; i >= 0; i--, j++)  //Storing Elements of row (m - 1), from Columns (n - 2)
          to 1
         bounds[j] = a[m - 1][i];
      for(i = m - 2; i >= 1; i--, j++)  //Storing Elements of Column 0, from rows (m - 2) to 1
         bounds[j] = a[i][0];

      for(i = 0; i < bounds.length; i++)    //Sorting the array of boundary elements
         for(j = 0; j < bounds.length - 1; j++)
            if(bounds[j] > bounds[j+ 1])
            {
               int temp = bounds[j];
               bounds[j] = bounds[j + 1];
               bounds[j + 1] = temp;
            }

      for(i = 0, j = 0; i < n; i++, j++)   //Filling elements in row 0
         a[0][i] = bounds[j];
      for(i = 1; i < m - 1; i++, j++)   //Filling elements in column (n - 1), from row 1 to row
          (m-2)
         a[i][n - 1] = bounds[j];
      for(i = n - 1; i >= 0; i--, j++)   //Filling Elements in row (m - 1), from Columns (n -
          2) to 1
         a[m - 1][i] = bounds[j];
      for(i = m - 2; i >= 1; i--, j++)   //Filling Elements in Column 0, from rows (m - 2) to 1
```

```
      a[i][0] = bounds[j];

   System.out.println("Rearranged Matrix:");
   for(i = 0; i < m; i++)
   {
      for(j = 0; j < n; j++)
         System.out.print(a[i][j] + "\t");  //Printing the Rearranged Array
      System.out.println();
   }


   System.out.println("Only Boundary Elements: ");
   for(i = 0; i < m ; i++)          //Displaying only boundary elements
   {
      for(j = 0; j < n; j++)
      {
         if(i > 0 && i < m - 1 && j > 0 && j < n - 1)
            System.out.print("\t");
         else
            System.out.print(a[i][j] + "\t");
      }
      System.out.println();
   }

   for(i = 0; i < bounds.length; i++)  //Calculating sum of boundary elements
      sum += bounds[i];

   System.out.println("Sum Of Boundary Elements = " + sum);
  }
}
```

5 Date In Words

Take user input for a date in ddmmyyyy format and check its validity. Display the date in words if it is valid.

Input:
12052013

Output:
12th May, 2013

5.1 Algorithm

1) Take user input for the date in the "ddmmyyyy" format

2) Declare and initialize 3 StringBuffer objects passing the input string as a parameter.

3) Set the length of the first StringBuffer object to 2 to extract the first two characters in the String i.e. the date

4) Set the length of the second StringBuffer object to 4, and reverse it to eliminate the year part. After reversing, set the length to 2 and reverse again to eliminate the date part.

5) Reverse the third StringBuffer object, set the length to 4 and reverse again to extract the year.

6) Parse the data from the StringBuffer objects into three int varibles (date, month, year)

7) Check the validity of dates, with conditions related to date, month, and number of days per month, also checking for leap year. Display appropriate message if an invalid date in entered.

8) Print the date along with an appropriate suffix by using a switch case

9) Declare a String array month[ ] and print out the month using the (n-1)th index. Print out the year as it is.

5.2 Code

```
import java.util.Scanner;
public class DateInWords
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter Date in format: ddmmyyyy");
      String input = s.next();   //Takes User input as a String

      StringBuffer p1, p2, p3;   //Declaration and initialisation of three String Buffers with
          the input String
      p1 = new StringBuffer(input);
      p2 = new StringBuffer(input);
      p3 = new StringBuffer(input);

      p1.setLength(2);      //First two characters extracted

      p2.setLength(4);      //Middle two characters extracted
      p2.reverse();
      p2.setLength(2);
      p2.reverse();
```

```
      p3.reverse();         //Last two characters extracted
      p3.setLength(4);
      p3.reverse();

      int date = Integer.parseInt(p1.toString());  //StringBuffers parsed to Integers
      int month = Integer.parseInt(p2.toString());
      int year = Integer.parseInt(p3.toString());

      if((date > 30 && (month == 2 || month == 4 || month == 6 || month == 9 || month == 1)) ||
          date > 31 ||  //Checking Validity of Dates
           (date > 28 && month == 2 && year % 4 != 0) || (date > 29 && month == 2 && year % 4
               == 0))
         System.out.println("Invalid Date");
      else
      {
         System.out.print(date);  //Date along with appropriate suffix printed
         switch(date)
         {
            case 1: System.out.print("st "); break;
            case 2: System.out.print("nd "); break;
            case 3: System.out.print("rd "); break;
            case 21: System.out.print("st "); break;
            case 22: System.out.print("nd "); break;
            case 23: System.out.print("rd "); break;
            case 31: System.out.print("st "); break;
            default: System.out.print("th "); break;
         }

         String months[] = {"January", "February", "March", "April", "May", "June", "July",
             "August", "September", "October", "November", "December"}; //Array of Month Names
         System.out.print(months[month - 1] + ", " + year);   //Printing of Year and Month
      }
   }
}
```

6 Determinant Solver

Write a program to solve a determinant of order 'n' recursively

Input:

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

Output:
Solution: -52

6.1 Algorithm

1) Take user input for order of the determinant and check its validity.

2) Declare a 2-D array (det[ ][ ]) of the given order.

3) Take user input for the elements using two nested loops.

4) Create an object of the DeterminantSolver class (d) and call the Solve(int[ ][ ] det) method, passing det as the parameter.

5) Store the value of Solve() in a variable and print it out as the solution.

int Solve(int[ ][ ] det):

1) Declare an int 'rows' and set it equal to det.length

2) Check if rows = 1

A) Return det[0][0] (Only element in the determinant)

3) Declare an int 'cols' and set it equal to rows.

4) Declare an int 'val' and set it equal to 0.

5) Run a loop from a = 0 to a < cols, incrementing a at each iteration.

A) Declare a 2-D int array newDet[ ][ ] and set order equal to rows - 1.

B) Run a loop from i = 1 to i < rows, incrementing i at each iteration.

i) Run a loop from j = 0 to j < cols, incrementing j at each iteration

a) Check if j < a,

I) set newDet[i - 1][j] to det[i][j]

b) Check if j > a

I) set newDet[i - 1][j - 1] to det[i][j]

C) set val to val + det[0][a]$*(-1^a)*$Solve(newDet)

6) Return val

6.2 Code

```
import java.util.Scanner;

public class DeterminantSolver
{
```

```
  public static void main(String[] args)
  {
     Scanner s = new Scanner(System.in);

     System.out.println("Enter Order of Determinant");
     int n = s.nextInt();  //Takes input for order of the Determinant

     if(n <= 0)            //Checks validity of input
     {
        System.out.println("Invalid Size. Program will Exit");
        System.exit(0);
     }

     int[][] det = new int[n][n]; //Declares a determinant of given order

     System.out.println("Enter Elements, row-wise"); //Loop for entry of data into array
     for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
           det[i][j] = s.nextInt();

     DeterminantSolver d = new DeterminantSolver(); //Object of same class
     int sol = d.Solve(det);  //Passes determinant into recursive function Solve(int[][] det)

     System.out.println("\n\n Solution: " + sol);
  }

  public int Solve(int[][] det)
  {
     int rows = det.length;  //Parses number of rows
     if(rows == 1)         //Returns sole element if order is 1
        return det[0][0];
     int cols = rows;

     int val = 0; //To store value of row in consideration

     for(int a = 0; a < cols; a++) //Runs a loop iterating through the columns
     {
        int[][] newDet = new int[rows - 1][cols - 1]; //Declares a new determinant of order
            one less than original passed into function

        for(int i = 1; i < rows; i++) //Runs loop through the rows
        {
           for(int j = 0; j < cols; j++) //Runs loop through the columns
           {
              if(j < a)             //Skips the row and column in consideration in first loop
                 newDet[i - 1][j] = det[i][j];
              if(j > a)
                 newDet[i - 1][j - 1] = det[i][j];
           }
        }
        val += det[0][a] * ((int) (Math.pow(-1, a)) * Solve(newDet));  //Adds the value of
            current element to val
     }
     return val;  //Returns value of determinant
  }
}
```

7 PseudoArithmetic Series

Write a program to check for a PseudoArithmetic Series and display its sum.
A PseudoArithmetic series is one where elements from the opposite ends add up to a common sum.

Input:
2, 5, 7, 9, 12

Output:
It is a Pseudo Arithmetic Series
Common sum is 14
Total Sum is 42

7.1 Algorithm

1) Take user input of an array(A[ ]) or use Sample Data (Sample data used here)

2) Declare an int(sum) and store sum of first and last elements of array in it

3) Declare a flag for checking if it is a PseudoArithmetic Series or not

4) Run a loop from i = 0 and j = length - 1 till i $\leq$ j, incrementing i and decrementing j by 1 at each iteration

A) If A[i] + A[j] $\neq$ sum, raise a flag

5) Declare an int(num) to store half the number of elements in case of even number, (Length + 1) / 2 in case of odd

6) If it is a PseudoArithmetic Series, Display message, common sum and total sum

7.2 Code

```
public class PseudoArithmeticSeries
{

   public static void main(String[] args)
   {
      int[] A = {2, 5, 7, 9, 12}; //Sample Data

      int sum = A[0] + A[A.length - 1];
      boolean flag = true;


      for(int i = 0, j = A.length - 1; i <= j; i++, j--) //Checks oposite elements and sums them
         if((A[i] + A[j]) != sum) //If sums don't match
            flag = false;

      int num = (A.length % 2 == 0) ? A.length / 2 : (A.length + 1) / 2; //Stores number of
          elements in half the list for calculating sum

      if(flag)
         System.out.println("It is a Pseudo Arithmetic Series \nCommon sum is " + sum +
             "\nTotal Sum is " + (sum * num));
      else
         System.out.println("It is not a Pseudo Arithmetic Series");
   }
```

```
}
```

8 Harshad Number

Harshad Number is an integer (in base 10) that is divisible by the sum of its digits.

Input:
18

Ouput:
18 is a Harshad Number

8.1 Algorithm

1) Declare and Initialize a Scanner object to take user input

2) Take user input for a number and store it in an int (n)

3) Declare an int (sum) and initialize it to 0 to store sum of digits of the number

4) Run a loop from i = n to 0, Adding i

5) if the number (n) is divisible by the sum, It is a Harshad Number.

6) Display appropriate message

8.2 Code

```
import java.util.Scanner;
public class HarshadNum
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter Number");
      int n = s.nextInt(); //user Input of Number
      int sum = 0;  //To store the sum of digits

      for(int i = n; i > 0; sum += i % 10, i = i / 10) //Loop to obtain sum of digits
      {}

      if(n % sum == 0) //If the Number is divisible by its Sum
         System.out.println(n + " is a Harshad Number");
      else
         System.out.println(n + " is not a Harshad Number");
   }

}
```

9 Kaprekar Number

A positive whole number 'n' that has 'd' number of digits is squared and split into two pieces, a right-hand piece that has 'd' digits and the left-hand piece that has the remaining 'd' or 'd-1' digits. If the sum of the two pieces is equal to the number, then 'n' is a Kaprekar Number.

Input:
45

Output:
It is a Kaprekar Number

$45^2 = 2025$

$20 + 25 = 45$

9.1 Algorithm

1) Take user input into an int (inp)

2) Square the input and store it in another int (num)

3) Make a new StringBuffer (st) and set it to the square (num)

4) Store length of input into an int (d)

5) Declare a new StringBuffer (part1) and set to st and then set length to st.length() - d

6) Declare a new StringBuffer (part2) and set to st

7) Reverse part2, set length to d, reverse it again

8) Check if Integer values of part1 + part2 equals input, display appropriate output

9.2 Code

```
import java.util.Scanner;
public class KaprekarNum
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter A Number");
      int inp = s.nextInt();  //User input of integer
      int num = inp * inp;  //input squared

      StringBuffer st = new StringBuffer(Integer.toString(num)); //String Buffer initialised
          with String
      int d = Integer.toString(inp).length();  //length of original integer

      StringBuffer part1 = new StringBuffer(st.toString());
      part1.setLength(st.length() - d); //StringBuffer divided into parts of d and (length - d)
          characters


      //Making the second part of length (st.length - d)
      StringBuffer part2 = new StringBuffer(st.toString());
      part2.reverse();
      part2.setLength(d);
      part2.reverse();
```

```
      if(Integer.parseInt(part2.toString()) + Integer.parseInt(part1.toString()) == inp)
          //checking condition for Kaprekar Numbers
         System.out.println("It is Kaprekar Number");
      else
         System.out.println("It is not Kaprekar Number");

   }

}
```

10 Mobius Function

Mobius Function M(n) for a natural number n is defined as:

1) M(n) = 1 if n = 1

2) M(n) = 0 if any prime factor of N is contained in n more than once

3) M(n) = (-1)p if n is a product of p distinct prime factors

Input:
78

Output:
-1

$$78 = 2 * 3 * 13$$
$$M(78) = (-1)^3 = -1$$

10.1 Algorithm

1) Declare a Scanner object to take user input

2) Take user input and store it in an int (n)

3) Call primeFac(), passing n as an argument and print its value

void primeFac(int n):

1) Declare an int (primeCount) to store number of prime factors and initialize it to 0

2) If Number is Prime (using checkPrime()), return -1

3) Run a loop from i = 2, to n, incrementing i at each iteration

A) if i is Prime

i) if n is divisible by i more than one time (two times here), return 0

ii) if n is divisible by i one time, increment primeCount

4) return $(-1)^{primeCount}$

boolean checkPrime(int n):

1) if n is 0 or 1, return false

2) if n is 2, return true

3) if n is a multiple of 2, return false

4) Run a loop from 3 to sqrt(n) and increment by 2 at each iteration

A) If n is divisible by the current index, return false, else continue loop

5) return true

10.2 Code

```
import java.util.Scanner;
public class MobiusFn
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in); //To take user input
      MobiusFn f = new MobiusFn(); //Class Object

      System.out.println("Enter number to check:");
      int n = s.nextInt(); //Number to check
      s.close();

      System.out.println(f.primeFac(n)); //Prints value of M(n)
   }

   int primeFac(int n) //Function to Factorize the number
   {
      int primeCount = 0; //Number of Prime Factors

      if(checkPrime(n)) //Checks if the number is prime (has only 1 prime factor)
         return -1;

      for(int i = 2; i < n; i++) //Loops through numbers till n to check for factors contained
          more than once
      {
         if(checkPrime(i)) //Checks if the number is a prime
         {
            if(n % (i * i) == 0) //factor present more than once
               return 0;
            if(n % i == 0) //Factor present only once
               primeCount++;
         }
      }

      return (int) Math.pow(-1, primeCount); //returns (-1)^p
   }

   boolean checkPrime(int num) //Function to check if number is prime
   {
      if(num == 1 || num == 0)
         return false;
      if(num == 2)
         return true;
      if(num % 2 == 0)
         return false;

      for(int i = 3; i * i < num; i += 2)
         if(num % i == 0)
            return false;

      return true;
   }

}
```

11 GCD and LCM using Recursion

Write a program to find the GCD and LCM of two numbers using recursion

Input:
6, 4

Output:
GCD = 2
LCM = 12

11.1 Algorithm

1) Take inputs of two numbers(n, m) from the user

2) Print the value of GCD(n, m)

3) Print the value of LCM (m * n / GCD)

int GCD(int n, int m):

1) if m = 0, return n

2) Call GCD passing m and n % m as arguments and return its value

11.2 Code

```
import java.util.Scanner;
public class RecursiveGCD
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter two numbers:"); //Data Entry
      int n = s.nextInt();
      int m = s.nextInt();

      System.out.println("GCD = " + GCD(n, m));
      System.out.println("LCM = " + m * n / GCD(m, n)); //GCD * LCM = m * n
   }

   static int GCD(int n, int m) //Recursive Function to calculate GCD
   {
      if(m == 0)
         return n;
      return GCD(m, n % m);
   }

}
```

12 Note Dispenser

Write a program to take input of 10 amounts from the user and write them to a .DAT file. Read the file with the numbers and dispense notes and print the results.

Input (For one amount):
98

Output:
Amount: 98
1000: 0 notes
500: 0 notes
100: 0 notes
50: 1 notes
20: 2 notes
10: 0 notes
5: 1 notes
2: 1 notes
1: 1 notes

12.1 Algorithm

1) Declare an int array (denominations[ ]) and set it to available note denominations.

2) Create a new NoteDispense object.

3) Create a FileOutputStream (fis) object passing "NoteDispense.dat" as the parameter.

4) Create a DataOutputStream (dis) object and pass FileOutputStream object as parameter.

5) Declare an int array of length 10 (nums[ ]) to store the numbers.

6) Take user input for the number and check validity of the inputs.

7) Write the numbers to the file using dis.

8) Close all objects excluding NoteDispense.

9) Declare FileInputStream (fr) and DataInputStream (dr) objects with "NoteDispense.dat" and fr as parameters respectively.

10) Run an infinite for loop

   A) try calling dispense(), passing read line as the parameter.

   B) catch Exception and break loop if caught.

11) Close all objects.

void dispense(int n):

1) Declare an int x and store the passed value 'n' in it.

2) Declare an int array (noteNumber[ ]) of the same length as denominations and initialize all elements to 0.

3) Run a loop from i = 0 to 9, incrementing i at each iteration

   A) set noteNumber[i] to noteNumber[i] + n / denominations[i].

   B) set n to n % denominations[i]

4) Print amount and number of notes per denomination.

12.2 Code

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Scanner;

public class NoteDispense
{

   int[] denominations = {1000, 500, 100, 50, 20, 10, 5, 2, 1}; //Array for denominations

   public static void main(String[] args) throws Exception
   {
      Scanner s = new Scanner(System.in);
      NoteDispense nd = new NoteDispense();

      FileOutputStream fis = new FileOutputStream("NoteDispense.dat");  //Creates a .dat file
          to write to
      DataOutputStream dis = new DataOutputStream(fis);          //Passes object to D.O.S. to
          allow writing to the file

      int[] nums = new int[10];  //Array to store 10 Numbers to parse

      System.out.println("Enter 10 Amounts");
      for(int i = 0 ; i < 10; i++)  //Loop to input numbers into array
      {
         int n = s.nextInt();
         if(n < 0)               //Checks validity of input
         {
            System.out.println("Invalid Input");
            System.exit(0);
         }
         nums[i] = n;
      }

      for(int i = 0; i < 10; i++)      //Writes the numbers to the file
         dis.writeInt(nums[i]);

      fis.close();
      dis.close();       //fis, dis closed to save file
      s.close();

      FileInputStream fr = new FileInputStream("NoteDispense.dat");  //Input stream Objects
          declared for reading from the file
      DataInputStream dr = new DataInputStream(fr);

      for(;;)
      {
         try
         {
            nd.dispense(dr.readInt());    //reads numbers from the file and passes them to the
                dispense() function
         }
         catch(Exception e)              //Catches EndOfFile Exception and breaks the loop
         {
            break;
         }
      }
```

```
      dr.close();
      fr.close();
   }


   public void dispense(int n)   //Function to dispense notes
   {
      int x = n;    //Saves the value in a varible

      int[] noteNumber = new int[denominations.length];  //Array to store number of notes
      for(int i = 0; i < noteNumber.length; i++)       //Loop to initialize the new array
         noteNumber[i] = 0;

      for(int i = 0; i < 9; i++)  //Loop to iterate and calculate number of notes for each
          denomination
      {
         noteNumber[i] += (n / denominations[i]);  //Adds integral multiples of the current
             denomination to the number
         n = n % denominations[i]; //Sets value to remainder
      }

      System.out.println("Amount: " + x);
      for(int i = 0; i < noteNumber.length; i++)  //Prints Amount and the denominations
      {
         System.out.println(denominations[i] + ": " + noteNumber[i] + " notes");
      }

      System.out.println("\n\n");
   }
}
```

13 Numbers to Words

Write a prgram to take input of a number between 0 and 1000 and print it out in words.

Input:
217

Output:
Two Hundred Seveteen

13.1 Algorithm

1) Declare String array s_units[ ] and store number names from 1-9 with an empty string at the first index (index 0) to correspond with the indices.

2) Declare String array s_tens[ ] and store number names for tens digits (twenty, thirty, forty, ... ninety) with two empty strings in the first two indices (Indices 0 and 1).

3) Declare String array s_Teens[ ] and store number names for numbers from 10-19 corresponding to indices.

4) Take input of number and check its validity and display appropriate message for invalid input.

5) Parse the String into a StringBuffer and reverse it and append 0 till the StringBuffer is of length 3.

6) Reverse the String again and convert the StringBuffer to a Character array digits[ ].

7) If the number contains digit at the hundred's place and is not followed by 10-19, Print the number name of the digit at hundred's place from array s_units followed by "Hundred", the digit at Ten's place from array s_Tens and the units digit from array s_units.

8) If the number contains digit at the hundred's place and is followed by numbers 10-19, Print the number name of the digits at hundred's place from array s_units followed by "Hundred", the digits in the following two places from the array s_Teens.

9) If the number does not contain Hundred's Digit and the number does not belong to 10-19, Print the number at the ten's place from the array s_Tens followed by the number at unit's place from the array s_units.

10) If the number does not contain Hundred's digit and lies between 10-19, Print the number name from the array s_Teens.

13.2 Code

```
import java.util.Scanner;

public class NumToWords
{

   public static void main(String[] args)
   {
      String[] s_units = {"", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight",
          "Nine"}; //For storing number names of digits in the units place excluding those for
          10-19
      String[] s_Tens = {"", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy",
          "Eighty", "Ninety"}; //For storing number names of digits in the tens place excluding
          those for 10-19
```

```
    String[] s_Teens = {"Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen",
        "Sixteen", "Seventeen", "Eighteen", "Nineteen"}; //For storing number names of
        numbers in the range 10-19

    Scanner s = new Scanner(System.in);

    System.out.println("Enter Number between 0 and 1000 (Exclusive)");
    int n = s.nextInt();
    if(n <= 0 || n >= 1000)  //Checks validity of input
    {
       System.out.println("Value out of Range");
       System.exit(0);
    }

    StringBuffer st = new StringBuffer(Integer.toString(n)); //Converts Integer to a
        StringBuffer
    st.reverse(); //Reverses the String
    int l = st.length(); //Stores length of the String

    if(l < 3)  //If the length is less than maximum allowed, appends zeroes till length is at
        maximum
       for(int i = 0; i < 3 - l; i++)
         st.append(0);

    st.reverse(); //Reverses String again

    char digits[] = st.toString().toCharArray(); //Converts String to Character Array

    if(Character.getNumericValue(digits[1]) != 1 && Character.getNumericValue(digits[0]) !=
        0) //Contains Hundreds Digits and does not belong to 10-19
       System.out.println(s_units[Character.getNumericValue(digits[0])] + " Hundred " +
           s_Tens[Character.getNumericValue(digits[1])] + " " +
           s_units[Character.getNumericValue(digits[2])]);

    if(Character.getNumericValue(digits[1]) == 1 && Character.getNumericValue(digits[0]) !=
        0) // Contains Hundreds Digits and belongs to 10-19
       System.out.println(s_units[Character.getNumericValue(digits[0])] + " Hundred " +
           s_Teens[Character.getNumericValue(digits[2])]);

    if(Character.getNumericValue(digits[1]) != 1 && Character.getNumericValue(digits[0]) ==
        0) // Does not Contain Hundred's Digit and does not Belong to 10-19
       System.out.println(s_Tens[Character.getNumericValue(digits[1])] + " " +
           s_units[Character.getNumericValue(digits[2])]);

    if(Character.getNumericValue(digits[1]) == 1 && Character.getNumericValue(digits[0]) ==
        0) //Does Not Contain Hundred's Digit and Belongs to 10-19
       System.out.println(s_Teens[Character.getNumericValue(digits[2])]);
  }

}
```

14 Primes Array

Write a program to accept dimensions of an array and fill it with prime numbers.

Input:
Rows = 3
Columns = 3

Output:

```
 2   3   5
 7  11  13
17  19  23
```

14.1 Algorithm

1) Take input for number of rows and columns from the user and store in two variables (m, n)

2) Create a new 2-D int array (A) of size m * n

3) Create a new int array (primes) of size (m*n)

4) Declare an int (k) and initialize it to 2

5) Run a loop from i = 0 to m*n and increment k at each iteration

   A) Check if current value of k is a prime, if true, go to i)

      i) Set primes[i] to k

      ii) Increment i by 1

6) Set k to 0

7) Run a loop from i = 0 to m, incrementing i at each iteration

   A) Run a loop from j = 0 to n, incrementing j at each iteration

      i) Set A[i][j] to primes[k]

      ii) Increment k by 1

8) Print array A

boolean checkPrime(int n):

1) if n is 0 or 1, return false

2) if n is 2, return true

3) if n is a multiple of 2, return false

4) Run a loop from 3 to sqrt(n) and increment by 2 at each iteration

   A) If n is divisible by the current index, return false, else continue loop

5) return true

14.2 Code

```
import java.util.Scanner;
public class PrimesInArray
{
   public static void main(String[] args)
```

```
  {
     Scanner s = new Scanner(System.in);

     System.out.println("Enter Number of Rows");   //Take input for number of rows
     int m = s.nextInt();


     System.out.println("Enter Number of Columns");  //Take input for number of columns
     int n = s.nextInt();

     int[][] A = new int[m][n];               //Declare array of size of input values

     int i, j, k = 2;
     int[] primes = new int[m * n];             //Declare array of size m * n

     for(i = 0; i < m * n; k++)               //Run through numbers more than 2
     {
        if(checkPrime(k))                   //Check if current number is prime
        {
           primes[i] = k;
           i++;                            //Increment counter if number is prime
        }
     }

     k = 0;

     for(i = 0; i < m; i++)                   //Write primes to 2-D array
        for(j = 0; j < n; j++)
           {
              A[i][j] = primes[k];
              k++;
           }

     System.out.println("Your Prime Array");

     //Print Array

     for(i = 0; i < m; i++)
     {
        for(j = 0; j < n; j++)
           System.out.print(A[i][j] + "\t");

        System.out.println();
     }

  }

  public static boolean checkPrime(int n)  //Function to check Primes
  {
     if(n == 2)
        return true;
     if (n % 2 == 0)
        return false;
      for(int i = 3; i * i <= n; i += 2)
          if(n % i == 0)
              return false;

      return true;
  }
}
```

15 Quick Sort(Array)

Write a program to implement quicksorting in an array.

Input:
2 6 4 8 1

Output:
1 2 4 6 8

15.1 Algorithm

1) Declare an int array (A) and take user input

2) Call quicksort on entire array (0, A.length - 1)

3) Print array A

void quicksort(int[ ] A, int left, int right):

1) Declare an int q

2) If right is more than left,

   A) Set q as Partition(A, left , right) [Partition of entire array]

   B) Call Quicksort on first half (left, q - 1)

   C) Call quicksort on second half (q + 1, right)

int partition(int[ ] A, int left, int right):

1) Declare an int P (Pivot element) and set it to the first element in the given array (A[left])

2) Delare an int i and set it to left, another int j and set it to right + 1

3) Run an infinite loop

   A) While A[++i] is less than Pivot element,

      i) Check if i $\geq$ right, if yes, break

   B) While A[−−j] is more than Pivot element

      i) Check if j $\leq$ left, if yes, break

   C) If i is $\geq$ j,

      i) break

   D) Else,

      i) swap I and j

4) if j = left,

   A) return j

5) swap elements at left and j

6) return j

void swap(int[ ] A, inti, int j):

Swap A[i] and A[j]

15.2 Code

```
import java.util.Scanner;
public class QuickSortArray
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter size of array");
      int n = s.nextInt();

      int A[] = new int[n];

      System.out.println("Enter elements");
      for(int i = 0; i < n; i++)
         A[i] = s.nextInt();

      QuickSortArray qs = new QuickSortArray();

      qs.quicksort(A, 0, A.length - 1);     //Calls quicksort on given array

      for(int i = 0; i < A.length; i++)
         System.out.print(A[i] + " ");
   }

   public void quicksort(int A[], int left, int right)
   {
      int q;
      if(right > left)           //if upper index is more than lower index
      {
         q = partition(A, left, right); //Calls partition on array between left and right
             indices
         quicksort(A, left, q - 1);  //Calls itself on array between left and index of j passed
             back from partition function
         quicksort(A, q + 1, right);   //Calls itself on array between q + 1 and right
      }
   }

   public int partition(int A[], int left, int right) //Function to split array into 3 parts,
       left block, pivot, right block
   {
      int P = A[left];   //Set Pivot P to the first element in the given array
      int i = left;      //first index in array
      int j = right + 1;    //last index + 1 in array

      for(;;)
      {
         while(A[++i] < P) //finds last element in line which is smaller than Pivot
            if(i >= right) //Breaks if index exceeds array range
               break;

         while(A[--j] > P) //Finds last element in line (from the rear) which is larger than
             Pivot
```

```
            if(j <= left) //Breaks if index is smaller than lowest index of array
               break;


         if(i >= j)       //Breaks if i and j are same, or i is larger
            break;
         else
            swap(A, i, j); //Swap elements at i and j to put elements smaller than pivot in 1st
                subblock and larger than pivot in second subblock
      }

      if(j == left)       //If j has gone down to lowest index, returns lowest index
         return j;

      swap(A, left, j);  //Swap lowest index and current index of j
      return j;          //Returns j
   }

   public void swap(int[] A, int i, int j)  //Function to swap elements at indices i and j
   {
      int temp = A[i];
      A[i] = A[j];
      A[j] = temp;
   }

}
```

16 String Reverse (Recursive)

Write a recursive program to reverse a given String and check if it is a palindrome.

Input:
Java Is Great

Output:
taerG sI avaJ
They are not palindromes

16.1 Algorithm

1) Declare a String (Str) and a StringBuffer (Revst) and initialize it to null ("")

2) Create an object of the Revstr class

3) Call getStr()

4) Call recReverse(), passing Str.length - 1 as parameter (for last character in String)

5) Call check()

void getStr():

1) Create a new Scanner object for user Input

2) Take user input of a String and store it in Str

void recReverse(int n):

1) if n = 0, Append the first character of the String to the StringBuffer

2) else

   A) Append the nth character of the String to the StringBuffer

   B) Pass n-1 in recReverse()

void check():

1) Print String and Reversed String

2) Check if both are equal

3) Display appropriate message

16.2 Code

```
import java.util.Scanner;

public class Revstr
{

   String Str;   //To Store the input String
   StringBuffer Revst = new StringBuffer(""); //For editing and storing edited String

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

```
  {
     wks3_Revstr r = new wks3_Revstr(); //Object of class
     //Method Calls
     r.getStr();
     r.recReverse(r.Str.length() - 1);
     r.check();
  }


  void getStr() //Function to take input
  {
     Scanner s = new Scanner(System.in);
     System.out.println("Enter the String to reverse");
     Str = s.nextLine();
  }

  void recReverse(int n) //Recursive Function to reverse the string
  {
     if(n == 0) //Appends first character
        Revst.append(Str.charAt(0));
     else
     {
        Revst.append(Str.charAt(n)); //Appends nth character
        recReverse(n - 1); //Recursive call
     }
  }

  void check()  //Function to check if the String is a palindrome
  {
     System.out.println("Original String: " + Str);
     System.out.println("Reversed String: " + Revst);
     if(Str.equalsIgnoreCase(Revst.toString()))
        System.out.println("They are palindromes");
     else
        System.out.println("They are not palindromes");

  }
}
```

17 Character Count Without Loop

Write a Program to find the frequency of occurence of a character in a String without using loops of any kind.

Input:
Have A Nice Day
Character: e

Output:
Frequency of e is 2

17.1 Algorithm

1) Take user input of a String

2) Take input for the character to look for

3) Split the String by the given character and store in an array

4) Frequency is given by length of the array - 1

17.2 Code

```
import java.util.Scanner;

public class CharCountWithoutLoop
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter String");
      String inp = s.nextLine();       //String to store input
      System.out.println("Enter character to find occurrence of");
      char c = s.next().charAt(0);     //character to search for

      String[] words = inp.split("" + c);  //splits String at each occurrence of character
      System.out.println("Frequency of " + c + " is " + (words.length - 1)); //Prints frequency
          of the character

      s.close();
   }

}
```

18 Special Numbers

Write a program to check if a given number is a Special Number.
A number is a Special number if the sum of the Factorials of it's digits equals the number itself.

Input:
145

Output:
145 is a Special Number
1! + 4! + 5! = 1 + 24 + 120 = 145

18.1 Algorithm

1) Take user input for a number

2) Declare an int(sum) to store the sums of the factorials of the numbers

3) Extract each digits from the number and find their factorials, adding each to sum

4) Check if the sum equals the number

int factorial(int n):

1) If n < 2, return 1

2) Else return n * factorial(n - 1)

18.2 Code

```
import java.util.Scanner;
public class SpecialNum
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter Number to check:");
      int n = s.nextInt(); //Data entry

      int sum = 0;  //To store sum of factorials

      String st = String.valueOf(n); //Converts to String
      for(int i = 0; i < st.length(); i++)
      {
         sum += factorial(Integer.parseInt(st.charAt(i) + "")); //Extracts each digit and adds
             its factorial
      }

      if(n == sum)
         System.out.println(n + " is a Special Number");
      else
         System.out.println(n + " is not a Special Number");
   }

   static int factorial(int n) //Recursive function to calculate factorial
   {
```

```
      if(n < 2)
         return 1;

      return n * factorial(n - 1);
   }
}
```

19 Saddle Point

Write a program to find the Saddle point of a matrix.
A saddle point is an element of the matrix such that it is the minimum element for the row to which it belongs and the maximum element for the column to which it belongs. Saddle point for a given matrix is always unique.

Input:
4 5 6
7 8 9
5 1 3

Output:
Saddle point = 7

19.1 Algorithm

1) Take user input for order of matrix (n) and declare an array of size n * n

2) Run loop from i = 0 to n, incrementing by 1 at each iteration

   A) Run loop from j = 0 to n, incrementing by 1 at each iteration

      i) Take input of element

      ii) Check for negative values

      iii) If positive, add input to array

3) Declare variables to store maximum and minimum (int) and a variable for flag and set it to 0

4) Run a loop from i = 0 to n , incrementing by 1 at each iteration

   A) Set min as A[i][0] (First element in the row), and x as 0

      i) Run a loop from j = 0 to n, incrementing by 1 at each iteration

      ii) If A[i][j] (Current Element) is less than min, set min to current element and x to j

   B) Set max to A[0][x] (First element in column where min was found)

   C) Run a loop from i = 0 to n, incrementing by 1 at each iteration

      i) If A[i][x] (Current element in column) is more than max, set max to current element

   D) If max is equal to min, print max value as saddle point and raise a flag

5) If flag = 0 (flag raised), Print saddle point not found

19.2 Code

```
import java.util.Scanner;

public class SaddlePoint
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter order of Matrix");
      int n = s.nextInt();
      int i = 0, j = 0, x = 0;
```

```
    int A[][] = new int[n][n]; //2-D array to store matrix

    System.out.println("Enter Elements of Array, Row-wise:");

    for(i = 0; i < n; i++)  //Loops to input data
    {
       for(j = 0; j < n; j++)
       {
          x = s.nextInt();
          if(x < 0)
          {
             System.out.println("Please enter only positive Integers");
             System.exit(0);
          }
          A[i][j] = x;
       }
    }

    int max, min, f = 0;
        for(i = 0; i < n; i++)  //Iterates through each row
        {
            min = A[i][0];      //Sets first element in the row as minimum
            x = 0;
            for(j = 0; j < n; j++) //Iterates through each column of the given row
            {
                 if(A[i][j] < min) //If current element is smaller than minimum, sets element
                      as minimum
                 {
                     min = A[i][j];
                     x = j;
                 }
            }


            max = A[0][x];         //Sets first element in the column where minimum was found
                 as maximum
            for(i = 0; i < n; i++)   //Iterates through column
            {
                 if(A[i][x] > max)    //if element is greater than max, sets new element as max
                 {
                     max = A[i][x];
                 }
            }

            if(max == min)         //If max and min are same, saddle point is found
            {
                System.out.println("Saddle point = " + max);
                f = 1;             //raises flag
            }

        }

        if(f == 0)             //If flag is not raised, saddle point does not exist
        {
            System.out.println("No saddle point");
        }
  }

}
```

20 Sieve Of Erasthosenes

Write a program to implement the Sieve of Erasthosenes to find all the prime numbers below a given limit and print them out, along with the number of primes.

Input:
Limit = 25

Output:
2 3 5 7 11 13 17 19 23
9 primes

20.1 Algorithm

1) Declare two int variables (maxNum and count) to store the Maximum limit to display primes up to and the count of primes in the given range, and take user input for maxNum.

2) Check for validity of Input, and display appropriate message for invalid input.

3) Declare a boolean array (primes[ ]) to implement the sieve.

4) Set all values in primes[ ] to true.

5) Set first element in primes (primes[0]) corresponding to the number '1' to false.

6) Run a loop to iterate through primes[ ] and do the following:

   A) Check if current element is true and is a prime number using isPrime() function, go to a) if true

      i) run a loop from (i+1) * 2 to maxNum, increment by (i+1) on each iteration and set each (j-1) element to false

7) Run a loop to iterate through primes[ ] and print (index + 1) as a numeric for all elements that are true and increment count with each print.

8) Print count to display total number of primes in the limit.

boolean isPrime(int n):

1) if n is 0 or 1, return false

2) if n is 2, return true

3) if n is a multiple of 2, return false

4) Run a loop from 3 to sqrt(n) and increment by 2 at each iteration

   A) If n is divisible by the current index, return false, else continue loop

5) return true

20.2 Code

```
import java.util.Scanner;

public class SieveOfErasthosenes
{

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

```
      Scanner s = new Scanner(System.in);  //Creates Object of Scanner

      System.out.println("Enter limit:");
      int maxNum = s.nextInt(), count = 0;

      if(maxNum <= 0)
      {
         System.out.println("Entered range is invalid");
         System.exit(0);
      }

      boolean[] primes = new boolean[maxNum];    //Creates boolean array to implement the sieve
      for(int i = 0; i < maxNum; i++)          //Sets all values in array to true
         primes[i] = true;

      primes[0] = false;                 //Sets 1 to false

      for(int i = 1; i < maxNum; i++)          //Loop to iterate through array
         if(primes[i] && isPrime(i + 1))    //Checks if current element is true and is a prime
            for(int j = 2 * (i + 1); j <= maxNum; j += (i + 1)) //Sets all multiples of the
                current prime element to false
               primes[j - 1] = false;

      for(int i = 0; i < maxNum; i++)  //Loop to iterate through the array
         if(primes[i])   //Prints the elemnt if it is prime
         {
            System.out.print((i + 1) + " ");
            count++;
         }

      System.out.println("\n" + count + " primes");  //Prints count of primes
   }

   public static boolean isPrime(int n)  //Function to check if a given number is Prime
   {
      if(n == 0 || n == 1)
         return false;
      if(n == 2)
         return true;
      if(n % 2 == 0)
         return false;
      for(int i = 3; i * i <= n; i += 2)
         if(n % i == 0)
            return false;
      return true;
   }

}
```

21 ASCII Decryption

Write a program to do the following:

A) String containing numbers is entered (has to be encoded)

B) String is reversed

C) Ascii values are obtained and converted to char values

D) Maximum ASCII value permitted is 122, if higher value is obtained, lower number of characters are taken

Input:
2312179862310199501872379231018117927

Output:
Have A Nice Day

21.1 Algorithm

1) Declare a Scanner object for user input

2) Take input String from the user

3) Declare a new StringBuffer object and initialize it to the input String

4) Reverse the StringBuffer and parse it to a String

5) Create a character array of the String

6) Loop through the character array from index 0 to length - 3 (no specific increment/decrement)

A) Extract three characters from the Char array into a String

B) If the parse Integer value of the String is more than 122,

i) Extract two characters from the Char array into a String

ii) Print the char value of the parsed Integer from the String

iii) Increment loop counter by 2

C) Else

i) Print the char value of the parsed Integer from the String

ii) Increment loop counter by 3

21.2 Code

```
import java.util.Scanner;

public class Weird_Decryption
{
   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter Encoded String:");
      String inp = s.next(); //Takes input String

      StringBuffer str = new StringBuffer(inp); //Parses to StringBuffer for ease in editing
      String rev = str.reverse().toString();  //Reverses the String
```

```
      char revChar[] = rev.toCharArray();   //Converts to a Character Array
      for(int i = 0; i < revChar.length - 3;)  //Runs till 3 characters left in string
      {
         String x = "" + revChar[i] + revChar[i + 1] + revChar[i + 2]; //Extracts 3 characters
         if(Integer.parseInt(x) > 122) //Parses to Integer, Checks if it is more than 122
         {
            x = "" + revChar[i] + revChar[i + 1]; //Takes only 2 characters
            System.out.print((char)Integer.parseInt(x)); //Prints character from ASCII Value
            i += 2;
         }
         else
         {
            System.out.print((char)Integer.parseInt(x)); //Prints character from ASCII Value
            i += 3;
         }
      }
   }

}
```

22 Merge Sort (Array)

Write a Program to sort an array using MergeSort

22.1 Algorithm

1) Declare an int array A and take user input

2) Call the MergeSort function, passing A, 0, A.length - 1 as argument

3) Print the Array

MergeSort(int[ ] A, int p, int r):

1) Declare a int variable to store the middle element (int q)

2) If p is more than r,

   A) Set q to average of p and r (middle element)

   B) Call merge sort on first half of array (p, q)

   C) Call merge sort on second half of array (q + 1, r)

   D) Call merge on entire array (p, q, r)

Merge(int[ ] A, int p, int q, int r):

1) Declare an int (n1) to store size of first half of array passed (p - q + 1)

2) Declare an int (n2) to store size of second half of array passed (r - q)

3) Declare an int array L of size n1 + 1, and array R of size n2 + 1

4) Fill first half of original array in L, and second half in R

5) Set final element in L and R to Integer.MAX_VALUE for comparison ease

6) Declare int variable (i and j) and initialize them to 0

7) Run a loop from k = p, to k $\leq$ r, incrementing k at each iteration

8) If L[i] $\leq$ R[j] (Comparing elements in the two arrays)

   A) Set A[k] to L[i] (Smaller element)

   B) Increment i by 1

9) Else

   A) Set A[k] to R[j]

   B) Increment j by 1

22.2 Code

```
import java.util.Scanner;

public class MergeSortArray
{

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

```
{
   Scanner s = new Scanner(System.in);

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

   int[] A = new int[n];

   System.out.println("Enter elements");
   for(int i = 0; i < n; i++)
      A[i] = s.nextInt();

   MergeSortArray ms = new MergeSortArray();
   ms.MergeSort(A, 0, A.length - 1);        //Passes the entire array to the function

   for(int i = 0; i < A.length; i++)
      System.out.print(A[i] + " ");
}

public void MergeSort(int[] A, int p, int r)
{
   int q;
   if(p < r)                        //Runs till lower index is less then upper index
   {
      q = (p + r) / 2;              //Gets middle index
      MergeSort(A, p, q);              //Passes first half of array to itself
      MergeSort(A, q + 1, r);          //Passes seconds half of aray to itself
      Merge(A, p, q, r);               //Calls merge on entire array with q as middle element
   }
}

public void Merge(int[] A, int p, int q, int r)
{
   int n1 = q - p + 1;   //Gets size of first half of array passed
   int n2 = r - q ;   //Gets size of second half of array passed
   int i, j, k;


   int[] L = new int[n1 + 1], R = new int[n2 + 1];  //Declaring arrays corresponding to
       halves

   //Filling elements in the new arrays

   for(i = 0; i < n1; i++)
      L[i] = A[p + i];
   for(j = 0; j < n2; j++)
      R[j] = A[q + j + 1];


   //Sets last element in array to Maximum value of integer for comparison
   L[n1] = Integer.MAX_VALUE;
   R[n2] = Integer.MAX_VALUE;

   i = 0;
   j = 0;

   for(k = p; k <= r; k++)  //Iterates thtough elements in the given range
   {
      //Rewrites elements to original array after comparison of two halves

      if(L[i] <= R[j])
```

```
        {
           A[k] = L[i];
           i++;
        }
        else
        {
           A[k] = R[j];
           j++;
        }
     }

  }
}
```

23 Phone Directory

Write a program to manage a Phone Directory using Files.

23.1 Algorithm

1) Declare class variables int count (to store number of records), String arrays (name, email, city, state, homeadd, workadd) int arrays (wrknum, homenum, mobnum, UID). Also declare objects which can be reused for ease.

2) Initialize a FileWriter Object passing the filename as the parameter and setting it to appendable so as to not overwrite and existing file but create a new one if it does not exist.

3) Close the FileWriter to save the new file.

4) Create a new PhoneDirectory Object, call menu().

void menu():

1) Initialize a Scanner object to take user input

2) Run a do while loop for the menu

   A) Initialize a FileWriter object with filename and make it appendable

   B) Initialize a PrintWriter object with the FileWriter

   C) Print out menu choices

   D) Take user input for the choice and pass it to a switch block and call appropriate functions in each block.

3) Close all Writers and Scanner

void dataParse():

1) Initialize a FileWriter object with filename and make it appendable

2) Initialize a Scanner object to read the file, passing FileWriter as a parameter

3) Declare a StringTokenizer for operations on the lines read from file

4) Declare an int 'count' and initialize it to 0

5) While, the file has a line, get the next line and increase count by 1 each time

6) Initialize all data arrays to a length of count

7) Close the Readers and re-initialize the same way to reset

8) Run a loop, reading each line and splitting the data using a StringTokenizer at ',' and parse the data to the respective arrays

9) Close the readers

void modifyRecord():

1) Parse the data from the file

2) Initialize the Scanner to take user input

3) Take input for the name of the record to search for

4) Declare an int (index) to store index of record and a boolean (exists) to check if record exists

5) Run a loop till count, checking if name matches input, and raise a flag and set index to current iteration value

6) Display data choices to modify and use a switch case to take appropriate input and modify it in the respective array at the 'index'

7) Initialize a FileWriter object and pass the filename as a parameter. Also initialize a PrintWriter and pass the FileWriter as a parameter to write to the file

8) Write all data to the file including the modified record

9) Close the writers

void deleteRecord():

1) Parse the data from the file

2) Initialize the Scanner to take user input

3) Take input for the name of the record to search for

4) Declare an int (index) to store index of record and a boolean (exists) to check if record exists

5) Run a loop till count, checking if name matches input, and raise a flag and set index to current iteration value

6) Initialize a FileWriter and pass the filename as the parameter

7) Initialize a PrintWriter, passing the FileWriter as a parameter to allow writing to the file

8) Run a loop from 0 to count, printing all data to the file excluding the record to be deleted

9) Close the writers

void addRecord():

1) Parse the data from the file

2) Initialize a Scanner for user input, FileWriter to allow writing to the file and make the file appendable, and PrintWriter to write to the file

3) Sort the data by UID by calling sortUID()

4) If file has data (count != 0), set UID as the number after the last UID, else set it as 100000

5) Take input from the user for data and print to the file

6) Close the writers

void sortUID():

1) Parse the data from the file

2) Use Bubble Sort to sort the data according to UIDs

void sortAlpha():

1) Parse the data from the file

2) Use Bubble Sort to sort the data according to Name

void sortCity():

1) Parse the data from the file

2) Use Bubble Sort to sort the data according to City

void sortState():

1) Parse the data from the file

2) Use Bubble Sort to sort the data according to State

void display():

1) Run a loop from 0 to count

   A) Print all data of each record

void displayAlpha():

1) Sort data alphabetically by name using sortAlpha()

2) Run a loop from 0 to count

   A) Print all data of each record

void displayCity():

1) Sort data alphabetically by city using sortCity()

2) Run a loop from 0 to count

   A) Print all data of each record

void displayState():

1) Sort data alphabetically by state using sortState()

2) Run a loop from 0 to count

   A) Print all data of each record

23.2 Code

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.util.Scanner;
import java.util.StringTokenizer;

public class PhoneDirectory
{

   int count; //For number of records
   String[] name, email, city, state, homeadd, workadd; //Personal Detail Fields
   int[] wrknum, homenum, mobnum; //Contact Information fields
```

```
int[] UID; //ID Field
static String filename = "phoneDirectory.txt"; //Filename for ease

//Objects for reusability
FileWriter fw;
PrintWriter pw;
Scanner s;


public static void main(String[] args) throws Exception
{
   FileWriter fw = new FileWriter(filename, true); //Creates file if it doesn't exist, does
       nothing otherwise since it is closed immediately after
   fw.close();

   PhoneDirectory pd = new PhoneDirectory(); //Creating a new Class object and calling the
       menu() function
   pd.menu();
}

public void menu() throws Exception
{
   Scanner s = new Scanner(System.in);
   int choice = 1;

   do
   {
      fw = new FileWriter(PhoneDirectory.filename, true); //Makes the file appendable,
          creates it if it doesn't exist
      pw = new PrintWriter(fw); //Object to allow writing to file

      choice = 0;
      System.out.println("Enter choice. Any other number will exit the program.");
      System.out.println("1 - Add Records");
      System.out.println("2 - Modify Records");
      System.out.println("3 - Delete Records");
      System.out.println("4 - Display Alphabetically");
      System.out.println("5 - Display City-Wise");
      System.out.println("6 - Display State-Wise");

      choice = s.nextInt(); //Takes input for menu choice

      switch(choice) //Switch block for calling appropriate function according to choice
      {
         case 1:    addRecord();
                 break;

         case 2:    modifyRecord();
                 break;

         case 3:    deleteRecord();
                 break;

         case 4:    displayAlpha();
                 break;

         case 5:    displayCity();
                 break;

         case 6:    displayState();
                 break;
```

```
      }

      pw.close();
      fw.close();

   } while(choice > 0 && choice < 7);

   fw.close();
   pw.close();
   s.close();
}

public void dataParse() throws Exception
{
   FileReader fr = new FileReader(PhoneDirectory.filename);  //Opens the file to read
   Scanner st = new Scanner(fr); //Allows reading data
   StringTokenizer stt; //For operations
   count = 0;    //Count of records

   while(st.hasNextLine())  //Loops till file has a line and counts lines
   {
      st.nextLine();
      count++;
   }

   //Data arrays initialised
   name = new String[count];
   email = new String[count];
   city = new String[count];
   state = new String[count];
   homeadd = new String[count];
   workadd = new String[count];
   wrknum = new int[count];
   homenum = new int[count];
   mobnum = new int[count];
   UID = new int[count];;

   fr.close();
   st.close();
   //Readers closed to reset them

   //Readers reopened to read data
   fr = new FileReader(PhoneDirectory.filename);
   st = new Scanner(fr);

   for(int i = 0; i < count; i++) //Loop to parse all data in each line and assign it to the
       respective array
   {
      stt = new StringTokenizer(st.nextLine(), ",");
      UID[i] = Integer.parseInt(stt.nextToken());
      name[i] = stt.nextToken();
      email[i] = stt.nextToken();
      city[i] = stt.nextToken();
      state[i] = stt.nextToken();
      homeadd[i] = stt.nextToken();
      workadd[i] = stt.nextToken();
      homenum[i] = Integer.parseInt(stt.nextToken());
      wrknum[i] = Integer.parseInt(stt.nextToken());
      mobnum[i] = Integer.parseInt(stt.nextToken());
   }
   fr.close();
```

```
   st.close();
}

public void modifyRecord() throws Exception
{
   dataParse(); //Parses data in file

   s = new Scanner(System.in); //For user input

   System.out.println("Enter Name of the person to modify record for: ");
   String sname = s.nextLine(); //Takes record to search for
   int index = 0; //Index where record is found
   boolean found = false; //To check if record exists

   for(int i = 0; i < count; i++) //Loops till record is found or list ends
      if(sname.equalsIgnoreCase(name[i]))
      {
         index = i;
         found = true;
         break;
      }

   if(!found)
   {
      System.out.println("Record not found, Returning to menu");
      return;
   }

   System.out.println("Enter Appropriate choice to modify:");
   System.out.println("1 - Name");
   System.out.println("2 - Email ID");
   System.out.println("3 - City");
   System.out.println("4 - State");
   System.out.println("5 - Home Address");
   System.out.println("6 - Work Address");
   System.out.println("7 - Home Phone Number");
   System.out.println("8 - Work Phone Number");
   System.out.println("9 - Mobile Phone Number");

   int c = s.nextInt();
   s.nextLine();

   switch(c)  //Switch Block to accept input for new data according to choice
   {
      case 1: System.out.println("Enter New Name");
            name[index] = s.nextLine();
            break;

      case 2: System.out.println("Enter New Email ID");
            email[index] = s.nextLine();
            break;

      case 3: System.out.println("Enter new City");
            city[index] = s.nextLine();
            break;

      case 4: System.out.println("Enter new State");
            state[index] = s.nextLine();
            break;

      case 5: System.out.println("Enter new Home Address");
```

```
            homeadd[index] = s.nextLine();
            break;

      case 6: System.out.println("Enter new Work Address");
            workadd[index] = s.nextLine();
            break;

      case 7: System.out.println("Enter new Home Phone Number");
            homenum[index] = s.nextInt();
            break;

      case 8: System.out.println("Enter new Work Phone Number");
            wrknum[index] = s.nextInt();
            break;

      case 9: System.out.println("Enter New Mobile phone Number");
            mobnum[index] = s.nextInt();
            break;
   }

   //Objects to write to the file
   fw = new FileWriter(PhoneDirectory.filename); //Overwrites the original file
   pw = new PrintWriter(pw);

   for(int i = 0; i < count; i++) //Writes new data to the file
   {
      pw.print(UID[i] + ",");
      pw.print(name[i] + ",");
      pw.print(email[i] + ",");
      pw.print(city[i] + ",");
      pw.print(state[i] + ",");
      pw.print(homeadd[i] + ",");
      pw.print(workadd[i] + ",");
      pw.print(homenum[i] + ",");
      pw.print(wrknum[i] + ",");
      pw.print(mobnum[i]);
      pw.println();
   }

   //Closes the writers to save the file
   fw.close();
   pw.close();
}

public void deleteRecord() throws Exception
{
   dataParse();    //Parses data from the file

   s = new Scanner(System.in);   //For user input

   System.out.println("Enter Name of the person to modify record for: ");
   String sname = s.nextLine(); //Takes record to search for
   int index = 0; //Index of record
   boolean found = false; //Checks if record found

   for(int i = 0; i < count; i++)   //Loop to find record and store index if found
      if(sname.equalsIgnoreCase(name[i]))
      {
         index = i;
         found = true;
         break;
```

```
      }

   if(!found)
   {
      System.out.println("Record not found, Returning to menu");
      return;
   }

   fw = new FileWriter(PhoneDirectory.filename); //To allow writing to file
   pw = new PrintWriter(pw); //To write to file

   for(int i = 0; i < count; i++) //Loop to print data to file
   {
      if(i == index) //Skips if record matches index of record to be deleted
         continue;

      pw.print(UID[i] + ",");
      pw.print(name[i] + ",");
      pw.print(email[i] + ",");
      pw.print(city[i] + ",");
      pw.print(state[i] + ",");
      pw.print(homeadd[i] + ",");
      pw.print(workadd[i] + ",");
      pw.print(homenum[i] + ",");
      pw.print(wrknum[i] + ",");
      pw.print(mobnum[i]);
      pw.println();
   }

   //Writers closed
   fw.close();
   pw.close();

}

public void addRecord() throws Exception
{
   dataParse(); //Parses data from the file

   s = new Scanner(System.in);   //For user input
   fw = new FileWriter(PhoneDirectory.filename, true); //Makes file appendable and allows
       writing
   pw = new PrintWriter(pw); //To write to the file

   int code = 0; //UID for the new record

   sortUID(); //Sorts data by UID

   if(count != 0) //For adding a record to existing file
      code = UID[count - 1] + 1;

   else    //For record in new file
      code = 100000;

   pw.print(code + ","); //Prints code to file

   //To take input for data and print it to file
   System.out.println("Enter Name:");
   pw.print(s.nextLine() + ",");
   System.out.println("Enter Email ID:");
   pw.print(s.nextLine() + ",");
```

```
   System.out.println("Enter City:");
   pw.print(s.nextLine() + ",");
   System.out.println("Enter State:");
   pw.print(s.nextLine() + ",");
   System.out.println("Enter Home Address (Separate Lines if applicable with hyphens(-) ):");
   pw.print(s.nextLine() + ",");
   System.out.println("Enter Work Address (Separate Lines if applicable with hyphens(-) ):");
   pw.print(s.nextLine() + ",");
   System.out.println("Enter Home Phone Number:");
   pw.print(s.nextInt() + ",");
   System.out.println("Enter Work Phone Number:");
   pw.print(s.nextInt() + ",");
   System.out.println("Enter Mobile Phone Number:");
   pw.println(s.nextInt());

   //Writers closed
   fw.close();
   pw.close();
}

public void sortUID() throws Exception //To sort the data according to UIDs
{
   dataParse();    //Parses data from the file

   for(int i = 0; i < UID.length; i++) //Sorts using Bubble Sort
   {
      for(int j = 0; j < UID.length-1; j++)
      {
         if(UID[j] > UID[j + 1])
         {
            String temp = name[j + 1];
            name[j + 1] = name[j];
            name[j] = temp;

            temp = email[j + 1];
            email[j + 1] = email[j];
            email[j] = temp;

            temp = city[j + 1];
            city[j + 1] = city[j];
            city[j] = temp;

            temp = state[j + 1];
            state[j + 1] = state[j];
            state[j] = temp;

            temp = workadd[j + 1];
            workadd[j + 1] = workadd[j];
            workadd[j] = temp;

            temp = homeadd[j + 1];
            homeadd[j + 1] = homeadd[j];
            homeadd[j] = temp;

            int temp2 = wrknum[j + 1];
            wrknum[j + 1] = wrknum[j];
            wrknum[j] = temp2;

            temp2 = homenum[j + 1];
            homenum[j + 1] = homenum[j];
            homenum[j] = temp2;
```

```
            temp2 = mobnum[j + 1];
            mobnum[j + 1] = mobnum[j];
            mobnum[j] = temp2;

            temp2 = UID[j + 1];
            UID[j + 1] = UID[j];
            UID[j] = temp2;
         }
      }
   }
}

public void sortAlpha() throws Exception  //To sort data Alphabetically by name
{
   dataParse(); //Parses data from the file

   for(int i = 0; i < name.length; i++) //Sorts data by Bubble Sort
   {
      for(int j = 0; j < name.length-1; j++)
      {
         if(name[j].compareToIgnoreCase(name[j + 1]) > 0)
         {
            String temp = name[j + 1];
            name[j + 1] = name[j];
            name[j] = temp;

            temp = email[j + 1];
            email[j + 1] = email[j];
            email[j] = temp;

            temp = city[j + 1];
            city[j + 1] = city[j];
            city[j] = temp;

            temp = state[j + 1];
            state[j + 1] = state[j];
            state[j] = temp;

            temp = workadd[j + 1];
            workadd[j + 1] = workadd[j];
            workadd[j] = temp;

            temp = homeadd[j + 1];
            homeadd[j + 1] = homeadd[j];
            homeadd[j] = temp;

            int temp2 = wrknum[j + 1];
            wrknum[j + 1] = wrknum[j];
            wrknum[j] = temp2;

            temp2 = homenum[j + 1];
            homenum[j + 1] = homenum[j];
            homenum[j] = temp2;

            temp2 = mobnum[j + 1];
            mobnum[j + 1] = mobnum[j];
            mobnum[j] = temp2;

            temp2 = UID[j + 1];
            UID[j + 1] = UID[j];
```

```
            UID[j] = temp2;
         }
      }
   }
}

public void sortCity() throws Exception  //To sort the data alphabetically by city
{
   dataParse(); //Parses data from the file

   for(int i = 0; i < city.length; i++) //Looop to sort data by bubble sort
   {
      for(int j = 0; j < city.length-1; j++)
      {
         if(city[j].compareToIgnoreCase(city[j + 1]) > 0)
         {
            String temp = name[j + 1];
            name[j + 1] = name[j];
            name[j] = temp;

            temp = email[j + 1];
            email[j + 1] = email[j];
            email[j] = temp;

            temp = city[j + 1];
            city[j + 1] = city[j];
            city[j] = temp;

            temp = state[j + 1];
            state[j + 1] = state[j];
            state[j] = temp;

            temp = workadd[j + 1];
            workadd[j + 1] = workadd[j];
            workadd[j] = temp;

            temp = homeadd[j + 1];
            homeadd[j + 1] = homeadd[j];
            homeadd[j] = temp;

            int temp2 = wrknum[j + 1];
            wrknum[j + 1] = wrknum[j];
            wrknum[j] = temp2;

            temp2 = homenum[j + 1];
            homenum[j + 1] = homenum[j];
            homenum[j] = temp2;

            temp2 = mobnum[j + 1];
            mobnum[j + 1] = mobnum[j];
            mobnum[j] = temp2;

            temp2 = UID[j + 1];
            UID[j + 1] = UID[j];
            UID[j] = temp2;
         }
      }
   }
}

public void sortState() throws Exception //To sort data alphabetically by state
```

```
{
   dataParse();//Parses data from the file

   for(int i = 0; i < state.length; i++) //Loop to sort data using Bubble Sort
   {
      for(int j = 0; j < state.length-1; j++)
      {
         if(state[j].compareToIgnoreCase(state[j + 1]) > 0)
         {
            String temp = name[j + 1];
            name[j + 1] = name[j];
            name[j] = temp;

            temp = email[j + 1];
            email[j + 1] = email[j];
            email[j] = temp;

            temp = city[j + 1];
            city[j + 1] = city[j];
            city[j] = temp;

            temp = state[j + 1];
            state[j + 1] = state[j];
            state[j] = temp;

            temp = workadd[j + 1];
            workadd[j + 1] = workadd[j];
            workadd[j] = temp;

            temp = homeadd[j + 1];
            homeadd[j + 1] = homeadd[j];
            homeadd[j] = temp;

            int temp2 = wrknum[j + 1];
            wrknum[j + 1] = wrknum[j];
            wrknum[j] = temp2;

            temp2 = homenum[j + 1];
            homenum[j + 1] = homenum[j];
            homenum[j] = temp2;

            temp2 = mobnum[j + 1];
            mobnum[j + 1] = mobnum[j];
            mobnum[j] = temp2;

            temp2 = UID[j + 1];
            UID[j + 1] = UID[j];
            UID[j] = temp2;
         }
      }
   }
}

public void display() //To display the data
{
   System.out.println("UID\t\tName\t\temail\t\tCity\t\tState\t\tWork Address\t\tHome
       Address\t\tWork Number\t\tHome Number\t\tMobile Number");
   for(int i = 0; i < count; i++)
   {
      System.out.print(UID[i] + "\t\t");
      System.out.print(name[i] + "\t\t");
```

```
            System.out.print(email[i] + "\t\t");
            System.out.print(city[i] + "\t\t");
            System.out.print(state[i] + "\t\t");
            System.out.print(workadd[i] + "\t\t");
            System.out.print(homeadd[i] + "\t\t");
            System.out.print(wrknum[i] + "\t\t");
            System.out.print(homenum[i] + "\t\t");
            System.out.println(mobnum[i] + "\t\t");
        }
    }

    public void displayAlpha() throws Exception //Displays data after sorting alphabetically by
        name
    {
        sortAlpha();
        display();
    }

    public void displayCity() throws Exception //Displays data after sorting Alphabetically by
        city
    {
        sortCity();
        display();
    }

    public void displayState() throws Exception //Displays data after sorting alphabetically by
        state
    {
        sortState();
        display();
    }

}
```

24 Student Records

Write a program to Manage Student Records and performing neccessary actions using Files.

24.1 Algorithm

1) Create class arrays for Name, Class, Roll Number, English, Math, Science, SS, Percent, Rank. Also make variables for linecount and studentcount.

2) Create object for FileWriter, passing the file as a parameter. Create a new StudentRecords object. Create a Scanner object to take user input.

3) Declare an int variable (menuChoice) to store choice for the menu.

4) Run a do while loop, while menuChoice lies in range.

   A) Display Menu options.

   B) Take user input for Menu.

   C) Pass the choice into a switch and call corresponding functions.

5) Close the objects (FileWriter and Scanner).

void dataEntry():

1) Create a FileWriter object, passing the filename as parameter along with a true to make the file appendable. pass this into a PrintWriter object as parameter.

2) Take input for number of students to enter data for and store it in an int (n).

3) Run a loop n times and take inputs for various data, printing it to the file.

4) Close the objects (FileWriter and PrintWriter)

void dataModify():

1) Call the dataParse() method to parse the file

2) Take user input for Roll number (key) of the student to modify data for

3) Declare a boolean flag (exists) and set it to false, to check if the record exists

4) Run a loop from i = 0 to stuCount, incrementing i by 1 at each iteration

   A) check if RollNo[i] is equal to key

      i) set exists to true

      ii) set an int position to i

5) If record does not exist

6) Display message and return to menu

7) Display all data of the student at position i and ask for confirmation

8) if confirmed, take input for new data, else return to menu

9) Reinitialise the FileWriter and PrintWriter passing the FileWriter as an argument for the PrintWriter. Make the file non-appendable to allow overwriting

10) Print all data to the file, overwriting previous data so changes are reflected.

11) Close FileWriter and PrintWriter.

void dataDelete():

1) Call the dataParse() method to parse the file

2) Take user input for Roll number (key) of the student to delete data for

3) Declare a boolean flag (exists) and set it to false, to check if the record exists

4) Run a loop from i = 0 to stuCount, incrementing i by 1 at each iteration

   A) check if RollNo[i] is equal to key

      i) set exists to true

      ii) set an int position to i

5) If record does not exist

   A) Display message and return to menu

6) Reinitialise the FileWriter and PrintWriter passing the FileWriter as an argument for the PrintWriter. Make the file non-appendable to allow overwriting

7) Print all data to the file, excluding the selected record, so that all data is overwritten with the new data excluding the record.

8) Close FileWriter and PrintWriter.

void displayAll():

1) Call the dataParse() method to parse the file

2) Run a loop from i = 0 to stuCount, incrementing by 1 at each iteration

   A) Print the student data for student at position i in all arrays excluding rank and percentage.

void displayMarksPercent():

1) Call the dataParse() method to parse the file.

2) Run a loop from i = 0, to stuCount, incrementing i at each iteration.

   A) Print Student Name, Marks in each subject, and percentage

void displayRank():

1) Call the dataParse() method to parse the file.

2) Run a loop from i = 0, to stuCount, incrementing i at each iteration

   A) Print Student Name, Percentage and Rank.

void dataParse():

1) Declare a FileReader instance, passing the filename as an argument Declare a Scanner instance, passing the FileReader as an arguemnt.

2) Declare an int (lineCount) and initialize it to 0.

3) Run a loop until the end of the file

   A) increment lineCount at each iteration

4) set stuCount to lineCount / 7

5) Initialise all data arrays to length equal to stuCount

6) Run a loop from i = 0, to stuCount, incrementing i at each iteration

A) Use the Scanner object to read lines from the file and parse the data into the respective arrays

B) Calculate average percentage of the given 4 subjects

7) Call the assignRank() function

8) Close Scanner and FileReader.

void assignRank():

1) Declare two int arrays Percentage[ ] and index[ ]. Set Percentage[ ] equal to the class variable Percent[ ], and initialize the index[ ] to a length of stuCount.

2) Run a loop from i = 0, to stuCount, incrementing i at each iteration

A) set index[i] to i.

3) Sort the Percentage array, and modify the index array at the same time to maintain the index number assigned to each record

4) Run a loop from i = 0, to stuCount, incrementing i at each iteration

A) Run a loop from j = 0, to stuCount, incrementing j at each iteration

i) if index[j] = i (compares index number with student number to assign rank)

a) Assign j+1 to rank[i]

24.2 Code

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.util.Scanner;

public class StudentRecords
{

   String fileName = "StudentRecords.txt"; //Filename for ease
   String Name[]; //Array for Student name
   int[] Class, RollNo, English, Math, Science, SS, Percent, Rank; //Arrays for Student data
   int lineCount, stuCount; //Other data for ease and reusability

   public static void main(String[] args) throws Exception
   {
      FileWriter fw = new FileWriter("StudentRecords.txt", true);  //Creates a new file which
          is appendable (or appends to existing file)
      Scanner s = new Scanner(System.in);
      StudentRecords stu = new StudentRecords();

      int menuChoice = 1;

      do
      {
         menuChoice = 0;
```

```
      System.out.println("Choose Operation\nEnter Any Other Number to Exit");
      System.out.println("1 - Enter Data");
      System.out.println("2 - Modify Data");
      System.out.println("3 - Delete Record");
      System.out.println("4 - Display All Data");
      System.out.println("5 - Display Marks and Percentage");
      System.out.println("6 - Display Rank");

      menuChoice = s.nextInt();

      switch(menuChoice)
      {
         case 1: stu.dataEntry();
               break;

         case 2: stu.dataModify();
               break;

         case 3: stu.dataDelete();
               break;

         case 4: stu.displayAll();
               break;

         case 5: stu.displayMarksPercent();
               break;

         case 6: stu.displayRank();
               break;
      }

   } while(menuChoice > 0 && menuChoice < 7);  //Displays menu until appropriate choice is
       given

   fw.close();
   s.close();
}

public void dataEntry() throws Exception //Function to input data
{
   FileWriter fw = new FileWriter(fileName, true);  //Creates object to allow appending to
       the file
   PrintWriter pw = new PrintWriter(fw);      //Object to allow printing to the file
   Scanner s = new Scanner(System.in);

   System.out.println("Enter Number of Students to Enter Data For:");
   int n = s.nextInt();  //To store number of students

   for(int i = 0; i < n; i++)  //Loop to take input for various data
   {
      System.out.println("Enter Data for Student #" + (i + 1) + ":");
      System.out.println("Name:");
      pw.println(s.nextLine());
      System.out.println("Class:");
      pw.println(s.nextInt());
      System.out.println("Roll Number:");
      pw.println(s.nextInt());
      System.out.println("English Marks (out of 100):");
      pw.println(s.nextInt());
      System.out.println("Math Marks (out of 100):");
```

```
      pw.println(s.nextInt());
      System.out.println("Science Marks (out of 100):");
      pw.println(s.nextInt());
      System.out.println("Social Studies Marks (out of 100):");
      pw.println(s.nextInt());
      System.out.println("\n\n");
   }

   fw.close();
   pw.close();
}

public void dataModify() throws Exception  //Function to modify entered data
{
   dataParse();    //Parses the file
   Scanner s = new Scanner(System.in);
   System.out.println("Enter Roll Number of Student to Modify Record for:");
   int key = s.nextInt(), position = 0;  //Stores Roll Number to modify data for
   boolean exists = false;  //Flag to check if record exists

   for(int i = 0; i < stuCount; i++)  //Iterates through Roll Number array
      if(RollNo[i] == key) //Checks if current element is the required record
      {
         exists = true;
         position = i;
      }

   if(exists == false)
   {
      System.out.println("Record Does Not Exist");
      return;
   }

   //Displays Data for Confirmation

   System.out.println("\n\n");
   System.out.println("Student Number: " + (position + 1));
   System.out.println("Student Name: " + Name[position]);
   System.out.println("Class: " + Class[position]);
   System.out.println("Roll Number: " + RollNo[position]);
   System.out.println("English: " + English[position]);
   System.out.println("Math: " + Math[position]);
   System.out.println("Science: " + Science[position]);
   System.out.println("Social Science: " + SS[position]);

   System.out.println("Are you sure you want to modify data for this student? (1 = yes, any
       other number = no");
   int confirm = s.nextInt(); //For confirmation

   if(confirm != 1)
      return;

   //Inputting new data

   System.out.println("Enter Data for Student #" + (position + 1) + ":");
   System.out.println("Name:");
   Name[position] = s.nextLine();
   System.out.println("Class:");
   Class[position] = s.nextInt();
   System.out.println("Roll Number:");
   RollNo[position] = s.nextInt();
```

```
   System.out.println("English Marks (out of 100):");
   English[position] = s.nextInt();
   System.out.println("Math Marks (out of 100):");
   Math[position] = s.nextInt();
   System.out.println("Science Marks (out of 100):");
   Science[position] = s.nextInt();
   System.out.println("Social Studies Marks (out of 100):");
   SS[position] = s.nextInt();

   FileWriter fw = new FileWriter(fileName); //Makes the file non-appendable to allow
       overwriting
   PrintWriter pw = new PrintWriter(fw);

   for(int i = 0; i < stuCount; i++) //Prints data to file from the start to overwrite
   {
      pw.println(Name[i]);
      pw.println(Class[i]);
      pw.println(RollNo[i]);
      pw.println(English[i]);
      pw.println(Math[i]);
      pw.println(Science[i]);
      pw.println(SS[i]);
   }

   fw.close();
   pw.close();
}


public void dataDelete() throws Exception  //Function to delete selected record
{
   dataParse();    //Parses data from file
   Scanner s = new Scanner(System.in);
   System.out.println("Enter Roll Number of Student to Delete Record for:");
   int key = s.nextInt();  //Stores roll number to find record for
   boolean exists = false;  //Flag to check if record exists

   for(int i = 0; i < stuCount; i++)  //Iterates through the roll number array
      if(RollNo[i] == key) //Checks if current element is the required record
         exists = true;

   if(!exists)
   {
      System.out.println("Record Does Not Exist");
      return;
   }

   FileWriter fw = new FileWriter(fileName); //Non-appendable writer to allow overwriting
   PrintWriter pw = new PrintWriter(fw);

   for(int i = 0; i < stuCount; i++)  //Overwrites data to file
   {
      if(RollNo[i] == key)       //Skips writing record to be deleted
         continue;


      pw.println(Name[i]);
      pw.println(Class[i]);
      pw.println(RollNo[i]);
      pw.println(English[i]);
      pw.println(Math[i]);
```

```
      pw.println(Science[i]);
      pw.println(SS[i]);
   }

   fw.close();
   pw.close();
}

public void displayAll() throws Exception  //Function to Display all records as in the file
{
   dataParse();    //Parses data from file
   for(int i = 0; i < stuCount; i++) //Iterates through the data arrays and prints them
   {
      System.out.println("Student Number: " + (i + 1));
      System.out.println("Student Name: " + Name[i]);
      System.out.println("Class: " + Class[i]);
      System.out.println("Roll Number: " + RollNo[i]);
      System.out.println("English: " + English[i]);
      System.out.println("Math: " + Math[i]);
      System.out.println("Science: " + Science[i]);
      System.out.println("Social Science: " + SS[i]);
      System.out.println("\n\n");
   }
}

public void displayMarksPercent() throws Exception  //Function to display essential details
    along with marks and percentage
{
   dataParse();    //Parses data from file
   for(int i = 0; i < stuCount; i++)  //Iterates through the data arrays and prints Name,
       Roll Number, marks and percentage
   {
      System.out.println("Student Number: " + (i + 1));
      System.out.println("Student Name: " + Name[i]);
      System.out.println("English: " + English[i]);
      System.out.println("Math: " + Math[i]);
      System.out.println("Science: " + Science[i]);
      System.out.println("Social Science: " + SS[i]);
      System.out.println("Percentage: " + Percent[i]);
      System.out.println("\n\n");
   }
}

public void displayRank() throws Exception  //Function to Display Student number, name,
    percentage and rank
{
   dataParse();    //Parses Data from file
   for(int i = 0; i < stuCount; i++) //Iterates through the data arrays and print Name, Roll
       number, percentage and rank
   {
      System.out.println("Student Number: " + (i + 1));
      System.out.println("Student Name: " + Name[i]);
      System.out.println("Percentage: " + Percent[i]);
      System.out.println("Rank: " + Rank[i]);
      System.out.println("\n\n");
   }
}

public void dataParse() throws Exception //Function to parse data from file and perform
    required calculations
{
```

```
   FileReader fr = new FileReader(fileName);  //File Reader
   Scanner s = new Scanner(fr); //To read data
   lineCount = 0; //Counts total lines for reference

   while(s.hasNextLine())  //Loops until it reaches end of file, gives line count
   {
      s.nextLine();
      lineCount++;
   }

   stuCount = lineCount / 7; //Calculates number of students based on number of lines in the
       file

   //Initialization of the arrays

   Name = new String[stuCount];
   Class = new int[stuCount];
   RollNo = new int[stuCount];
   English = new int[stuCount];
   Math = new int[stuCount];
   Science = new int[stuCount];
   SS = new int[stuCount];
   Percent = new int[stuCount];
   Rank = new int[stuCount];

   for(int i = 0; i < stuCount; i++)  //Reads lines and parses the data into arrays
   {
      Name[i] = s.nextLine();
      Class[i] = Integer.parseInt(s.nextLine());
      RollNo[i] = Integer.parseInt(s.nextLine());
      English[i] = Integer.parseInt(s.nextLine());
      Math[i] = Integer.parseInt(s.nextLine());
      Science[i] = Integer.parseInt(s.nextLine());
      SS[i] = Integer.parseInt(s.nextLine());
      Percent[i] = (English[i] + Math[i] + Science[i] + SS[i]) / 4; //Calculates percentage
   }

   assignRank();   //Assigns rank to the students based on percentage

   s.close();
   fr.close();
}

public void assignRank()   //Function to assign rank to the students based on percentage
{
   int[] Percentage = Percent, index = new int[stuCount];  //Percentage is a copy of
       Percent[], index stores index of students for reference

   for(int i = 0; i < stuCount; i++)  //Assigns index in order
      index[i] = i;

   //Sorting the two arrays

   for(int i = 0; i < stuCount; i++)
      for(int j = 0; i < stuCount - 1; j++)
         if(Percentage[j + 1] < Percentage [j])
         {
            int temp = Percentage[j];
            Percentage[j + 1] = Percentage[j];
            Percentage[j] = temp;
```

```
            temp = index[j];
            index[j + 1] = index[j];
            index[j] = temp;
         }

   //Compares index with student and assigns appropriate rank

   for(int i = 0; i < stuCount; i++)
      for(int j = 0; j < stuCount; j++)
         if(index[j] == i)
            Rank[i] = j + 1;

  }
}
```

25 Smart Substring

Write a program that takes maximum 30 characters from a string but without cutting the words.

Input:
Featuring stylish rooms and moorings for recreation boats, Room Mate Aitana is a designer hotel built in 2013 on an island in the IJ River in Amsterdam.

Output:
Featuring stylish rooms and

25.1 Algorithm

1) Take user input for a String(inp), and declare a String 'st'

2) Take user input for maximum characters

3) Split the input String from blank spaces and store in an array (words[ ])

4) Run a loop from st = "" and i = 0, till length of st + words[i]is less than or equal to n and append words[i] and a blank space to string and increment i at each iteration

5) if length of st is more than n

A) trim st

B) if length of st is more than n

i) String st by blank space again and store it in an array(words2[ ])

ii) set st to blank again ("")

iii) Run a loop from i = 0, till i < length of words2, incrementing i at each iteration

a) append words2[i] and blank space to st

iv) trim st

6) Print st

25.2 Code

```
import java.util.Scanner;

public class SmartSubstring
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter String");
      String inp = s.nextLine(), st;     //Stores input String

      System.out.println("Enter Max Chars");
      int n = s.nextInt();          //Stores maximum number of characters
      int i = 0;                    //iteration variable
      String[] words = inp.split(" ");  //Splits string into String array containing words
      for(st = ""; st.length() + words[i].length() <= n; st = st + words[i] + " ", i++)  //Loop
          to append each word to empty string till length of new string < n
      {}

      if(st.length() > n)      //Checks if length of new string > n
```

```
    {
       st.trim();         //trims extra spaces
       if(st.length() > n)   //Checks length again
       {
          String words2[] = st.split(" "); //Splits String into words
          st = "";                //sets st to ""
          for(i = 0; i < words2.length - 1; i++) //appends each word to st
             st = st + words2[i] + " ";
          st.trim(); //trims extra spaces
       }
    }

    System.out.println(st);  //Prints smart substring

    s.close();
  }

}
```

26 Duplicate Character Counter

Write a java program to find duplicate characters and their count in a given string.

Input:
Better Butter

Output:
B : 2
e : 3
t : 4
r : 2

26.1 Algorithm

1) Declare a Scanner object for user input

2) Take user input of a String and store it (inp)

3) Make a char array(chars[ ]) and an int array(freq[ ]) to store characters in the String and their frequency respectively, initializing them to the same length as the input.

4) Run a loop from i = 0 to length of input - 1, incrementing i at each iteration

A) Run a loop from j = 0 to length of chars[ ] - 1, incrementing j at each iteration

i) if chars[j] is null,

a) Set char[j] to the current character in String

b) Increase frequency of the character to 1

c) Break the loop

ii) if chars[j] is equal to current character in String

a) Increase frequency of the character by 1

b) Break the loop

5) Print all the duplicate characters and their frequency (omitting characters with frequency of 1)

26.2 Code

```
import java.util.Scanner;

public class DuplicateCharCount
{
    public static void main(String args[])
    {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter String");
      String inp = s.nextLine();      //Stores input String

      char[] chars = new char[inp.length()]; //char array of length = length of input to store
          each character
      int[] freq = new int[inp.length()];  //int array of length = length of input to store
          frequency of each character

      for(int i = 0; i < inp.length(); i++) //iterates through characters in string
      {
```

```
        for(int j = 0; j < chars.length; j++) //iterates through character array (chars)
        {
           if(chars[j] == Character.MIN_VALUE) //if chars[j] is null
           {
              chars[j] = inp.charAt(i);  //sets chars[j] to current character
              freq[j] = 1;            //increases frequency of that character to 1
              break;
           }
           if(chars[j] == inp.charAt(i))  //if character present in array
           {
              freq[j]++;              //increases frequency of character by 1
              break;
           }
        }
     }

     for(int i = 0; i < chars.length; i++)  //Prints duplicate characters and their frequency
          (freq > 1)
     {
        if(chars[i] != Character.MIN_VALUE && freq[i] != 1)
           System.out.println(chars[i] + " : " + freq[i]);
     }

     s.close();
   }
}
```

27 Symmetric Matrix check and Sum of Diagonals

Write a program to check if the given matrix is Symmetric or not, and find the sums of the left and right diagonals.
A square matrix is said to be Symmetric, if the element of the i th row and j th column is equal to the element of the j th row and i th column.

Input:
Order = 3

1	2	3
2	4	5
3	5	6

Output:
The Matrix is Symmetric.
Sum of Left Diagonal = 11
Sum of Right Diagonal = 10

27.1 Algorithm

1) Take input for order of Matrix and check its validity

2) Create a Matrix of appropriate order(A[ ][ ])

3) Take user input for each element in the Matrix

4) Run a loop to iterate through the Matrix checking that each A[i][j] = A[j][i], and displaying appropriate message

5) If Matrix is Symmetric

A) Run a loop from i = 0 and j = 0 till i < n and j < n, incrementing i and j by 1 at each iteration

i) Add each A[i][j] to an int(sumLeft)

B) Run a loop from i = n - 1 and j = 0 till i ≥ 0 and j < n, decrementing i and incrementing j at each iteration

i) Add each A[i][j] to an int(sumRight)

C) Print sumLeft and sumRight

27.2 Code

```
import java.util.Scanner;

public class Symmetric_DiagonalSum
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter Order of the Matrix");
      int n = s.nextInt(); //Input

      if(n < 2 || n > 10) //Validation of Input
      {
         System.out.println("Order of matrix cannot be less than 2 or more than 10");
         System.exit(0);
```

```
    }

    int i, j, A[][] = new int[n][n], sumLeft = 0, sumRight = 0; //Iteration variables,
        Matrix, Sums of Diagonals

    System.out.println("Enter Elements of the Matrix, Row-wise"); //Data Entry
    for(i = 0; i < n; i++)
       for(j = 0; j < n; j++)
       {
          A[i][j] = s.nextInt(); //User Input for each element
       }

    for(i = 0; i < n; i++)
       for(j = 0; j < n; j++)
          if(A[i][j] != A[j][i]) //Checks for elements on opposites of the Matrix
          {
             System.out.println("Not Symmetrical");
             System.exit(0);
          }
    System.out.println("Symmetrical");


    for(i = 0, j = 0; i < n && j < n; i++, j++) //Left Diagonal Sum
          sumLeft += A[i][j];

    for(i = n - 1, j = 0; i >= 0 && j < n; i--, j++) //Right Diagonal Sum
          sumRight += A[i][j];

    //Printing
    System.out.println("Sum of Left Diagonal = " + sumLeft);
    System.out.println("Sum of Right Diagonal = " + sumRight);
  }

}
```

28 Calendar Generator

Input:
Year: 2016
Month: January
First Day Of Month: Wednesday

Output:

January 2016						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

28.1 Algorithm

1) Declare a Scanner object to take user input

2) Take user inputs for Year, Month and First day of the month

3) Check if the year is a leap year and assign a value to boolean(isLeapYear)

4) Declare a 5x7 int array(cal[ ][ ]) to write the calendar to

5) Declare two ints (days and startPos) to store number of days in the month and starting position in the calendar respectively

6) Assign days its value based on Month and if it is a leap year or not

7) Assign startPos according to the day of the week (Starting from Sunday = 0)

8) Run a loop from i = 0 and count = 1, till count is $\leq$ days, incrementing i at each iteration

A) if i = 0 (First row)

i) Run a loop from j = startPos, till j $<$ 7 and count $\leq$ days, incrementing j and count at each iteration

a) set cal[i][j] to count

B) Else

i) Run a loop from j = 0, till j $<$ 7 and count $\leq$ days, incrementing j and count at each iteration

a) set cal[i][j] to count

9) Print the array in given format

28.2 Code

```
import java.util.Scanner;

public class CalendarGenerator
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
```

```
System.out.println("Enter Year");
int year = s.nextInt();          //To store year
System.out.println("Enter Month");
String month = s.next().trim();    //To store month
System.out.println("Enter First Day of Month");
String day = s.next().trim();   //To store first day of the month

boolean isLeapYear = (year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0)); //To
    check for leap year

int cal[][] = new int[5][7];     //Creates an Array to store the dates

int days, startPos = 0;          //Days to store number of days in month, startPos to
    store start position of dates in array

if(month.equalsIgnoreCase("January") || month.equalsIgnoreCase("March") ||
    month.equalsIgnoreCase("May") || month.equalsIgnoreCase("July") ||
    month.equalsIgnoreCase("August")
     || month.equalsIgnoreCase("October") || month.equalsIgnoreCase("December"))
         //Checks for months with 31 days
   days = 31;
else if(month.equalsIgnoreCase("February") && isLeapYear)  //Checks for February in Leap
    Year
   days = 29;
else if(month.equalsIgnoreCase("February") && !isLeapYear)  //Checks for February in
    non-Leap year
   days = 28;
else
   days = 30;    //else month has 30 days

switch(day.toLowerCase())  //To assign startPos based on Day
{
   case "sunday": startPos = 0; break;
   case "monday": startPos = 1; break;
   case "tuesday": startPos = 2; break;
   case "wednesday": startPos = 3; break;
   case "thursday": startPos = 4; break;
   case "friday": startPos = 5; break;
   case "saturday": startPos = 6; break;
}

for(int i = 0, count = 1; count <= days; i++) //Loop to fill dates in array
{
   if(i == 0)
      for(int j = startPos; j < 7 && count <= days; j++, count++)  //For 1st row
         cal[i][j] = count;
   else
      for(int j = 0; j < 7 && count <= days; j++, count++)  //For other rows
         cal[i][j] = count;
}


//Printing
System.out.println("-----------------------------------------------------");
System.out.println("\t\t " + month + " " + year + "");
System.out.println("-----------------------------------------------------");
System.out.println("Sun\tMon\tTue\tWed\tThu\tFri\tSat");

for(int i = 0; i < 5; i++)
{
   for(int j = 0; j < 7; j++)
```

```
        {
           if(cal[i][j] == 0)
              System.out.print("\t");
           else
              System.out.print(cal[i][j] + "\t");
        }
        System.out.println();

     }

     s.close();
  }

}
```

29 Matrix in Spiral Form

Write a program to output a given 2-D array in spiral form

Input:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Output:
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

29.1 Algorithm

1) Declare a Scanner object for user input

2) Take input for number of rows and columns and declare an array(a[ ]) of appropriate size

3) Declare variables for limits, keeping up and left as 0, down and right as rows - 1 and columns - 1 respectively

4) Declare an int elements = rows * columns

5) Take user input and add it to array

6) Print original array

7) Run a loop from i = 0, j = 0 till elements > 0, decrementing elements at each iteration

A) Print a[i][j]

B) if Direction is right

i) if j+1 > right limit

a) Set direction to down

b) Increase upper limit by 1

c) Increase row number

C) if Direction is down

i) if i+1 > down limit

a) Set direction to left

b) Decrease right limit by1

c) Decrease column number

D) if Direction is left

i) if j-1 < left limit

a) Set direction to up

b) Decrease down limit by 1

c) Decrease row number

E) if Direction is up

i) if i-1 < up limit

a) Set direction to right

b) Increase left limit by 1

c) Increase column number

29.2 Code

```
import java.util.Scanner;

public class MatrixInSpiral
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter No of Rows");
      int rows = s.nextInt();            //To store number of rows
      System.out.println("Enter No of Columns");
      int cols = s.nextInt();            //To store number of columns

      int[][] a = new int[rows][cols];   //new int array of size rows*columms

      String direction = "right";           //Direction to run loop in
      int up_limit = 0, down_limit = rows - 1, left_limit = 0, right_limit = cols - 1, elements
          = rows * cols; //Limits of each direction and total number of elements

      System.out.println("Enter data row-wise");
      for(int i = 0; i < rows; i++)      //Loop to input data
         for(int j = 0; j < cols; j++)
            a[i][j] = s.nextInt();

      System.out.println("Original 2D Array - ");
      for(int i = 0; i < rows; i++)      //Loop to print data
      {
         for(int j = 0; j < cols; j++)
            System.out.print(a[i][j] + "\t");

         System.out.println();
      }

      System.out.println("2D Array in Spiral Form - ");

      for(int i = 0, j = 0; elements > 0; elements--)  //Runs loop from number of elements to 0
          and row number(i) and column number(j)
      {
         System.out.print(a[i][j] + " ");      //Prints element at index (i, j)

         if(direction.equals("right"))         //For checking if current direction is 'right'
         {
            if(++j > right_limit)              //checks if j + 1 > limit of right
               {
                   --j;
                   direction = "down";            //direction changed to 'down'
                   ++up_limit;                    //upper limit increased
                   ++i;                       //row number increased
               }
         }
         else if(direction.equals("down"))        //For checking if current direction is 'down'
         {
```

```
            if(++i > down_limit)            //checks if i + 1 > limit of down
               {
                   --i;
                   direction = "left";           //direction changed to 'left'
                   --right_limit;                //right limit decreased
                   --j;                        //column number decreased
               }
         }
         else if(direction.equals("left"))       //For checking if current direction is 'left'
         {
            if(--j < left_limit)            //checks if j - 1 < limit of left
               {
                   ++j;
                   direction = "up";             //direction changed to 'up'
                   --down_limit;               //down limit decreased
                   --i;                        //row number decreased
               }
         }
         else if(direction.equals("up"))         //For checking if current direction is 'up'
         {
            if(--i < up_limit)                //checks if i - 1 < limit of up
               {
                   ++i;
                   direction = "right";          //direction changed to 'right'
                   ++left_limit;               //left limit increased
                   ++j;                        //column number increased
               }
         }
      }

      s.close();
   }

}
```

30 Circular Primes

A Circular Prime is a prime number that remains prime under cyclic shifts of its digits. When the leftmost digit is removed and replaced at the end of the remaining string of digits, the generated number is still prime. The process is repeated until the original number is reached again.

Input:
131

Output:
131
113
311

131 is Circular Prime

30.1 Algorithm

1) Declare a Scanner object to take user input

2) Take user input of an int(num)

3) Declare a boolean (prime) and set it to true. Declare a String (inp) and set it to num

4) Declare a StringBuffer(st), passing inp as argument

5) Run a loop from i = 0 to i < length of String, incrementing i at each iteration

   A) Print st

   B) Store the last character of st in a char(c)

   C) Remove the last character and append it at the front of st

   D) if the new number is prime

      i) continue loop

   E) else

      i) set prime to false

6) Print the appropriate message

boolean isPrime(int n):

1) if n is 0 or 1, return false

2) if n is 2, return true

3) if n is a multiple of 2, return false

4) Run a loop from 3 to sqrt(n) and increment by 2 at each iteration

   A) If n is divisible by the current index, return false, else continue loop

5) return true

30.2 Code

```
import java.util.Scanner;

public class CircularPrimeCheck
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter Number");
      int num = s.nextInt();          //Stores input
      boolean prime = true;           //Flag for prime

      String inp = num + "";          //Converts input to String


      StringBuffer st = new StringBuffer(inp); //StringBuffer used to rotate the String
      for(int i = 0; i < inp.length(); i++)  //Loop to rotate the String
      {
         System.out.println(st);         //Prints String

         char c = st.charAt(inp.length() - 1); //Character to store last char in current
             rotation of input String
         st.deleteCharAt(inp.length() - 1);  //Deletes saved character from string
         st.reverse();                //Reverses String
         st.append(c);                //Appends char
         st.reverse();                //Reverses String again

         if(isPrime(Integer.parseInt(st.toString()))) //Checks if current rotation is prime, if
             yes, goes to next rotation
            continue;
         else
            prime = false;                  //else, sets flag to false
      }

      System.out.println();
      if(prime)
         System.out.println(num + " is Circular Prime");
      else
         System.out.println(num + " is not Circular Prime");

      s.close();
   }

   public static boolean isPrime(int n)  //Function to check for prime
   {
      if(n == 0 || n == 1)
         return false;
      if(n == 2)
         return true;
      if(n % 2 == 0)               //Checks for multiple of 2
         return false;
      for(int i = 3; i * i < n; i += 2)  //Checks for multiples of all odd numbers
         if(n % i == 0)
            return false;
      return true;
   }

}
```

31 Character Types and Percentage

Write a program to find the number of each type of character in a String and calculate their respective percentages

Input:
Tiger Runs @ The Speed Of 100 km/hour.

Output:
Total Characters - 38.0
Uppercase Characters are 5 so, percentage = 13.157894736842104%
Lowercase Characters are 20 so, percentage = 52.63157894736842%
Numeric Characters are 3 so, percentage = 7.894736842105263%
Other Characters are 10 so, percentage = 26.31578947368421%

31.1 Algorithm

1) Declare a Scanner object to take user input

2) Take user input for a String(inp)

3) Declare double variables to store number of each type of Character

4) Run a loop from i = 0 to length of input - 1, incrementing i at each iteration

   A) Extract the current character in the String and store it in a char (c)

   B) Check the type of Character and increment the respective value by 1

5) Find total of characters and print it

6) Calculate percentage of each type of character and print them out

31.2 Code

```
import java.util.Scanner;

public class CharTypePercentage
{
   public static void main(String args[])
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter String");
      String inp = s.nextLine();       //String to store input

      double upper = 0, lower = 0, num = 0, other = 0; //variables to store number of
          occurrences of characters of each type

      for(int i = 0; i < inp.length(); i++)    //Loop to iterate through string
      {
         char c = inp.charAt(i);             //Character to store each character of String

         if(Character.isUpperCase(c))        //Checks if Character is Uppercase
            upper++;
         else if(Character.isLowerCase(c))     //Checks if Character is Lowercase
            lower++;
         else if(Character.isDigit(c))       //Checks if Character is Digit
            num++;
         else
```

```
            other++;                    //else it is Other Character
        }



        double total = upper + lower + num + other;  //Stores total number of characters
        System.out.println("Total Characters - " + total);

        double upperp = upper / total * 100;     //Percentage of Uppercase
        double lowerp = lower / total * 100;     //Percentage of Lowercase
        double nump = num / total * 100;       //Percentage of digits
        double otherp = other / total * 100;     //Percentage of Other Characters


        System.out.println("Uppercase Characters are " + (int)upper + " so, percentage = " +
            upperp + "%");
        System.out.println("Lowercase Characters are " + (int)lower + " so, percentage = " +
            lowerp + "%");
        System.out.println("Numeric Characters are " + (int)num + " so, percentage = " + nump +
            "%");
        System.out.println("Other Characters are " + (int)other + " so, percentage = " + otherp +
            "%");

        s.close();
    }
}
```

32 Reverse Each Word in a String

Write a program to reverse each word in a String preserving the original format of the String.

Input:
Have A Nice Day

Output:
evaH A eciN yaD

32.1 Algorithm

1) Declare a Scanner object to take user input

2) Take user input of a String(inp)

3) Split the string by blank space and store it in a String array(words[ ])

4) Declare an empty StringBuffer(st)

5) Run a loop from i = 0, to length of words[ ] - 1, incrementing i at each iteration

   A) Declare a StringBuffer(temp) and store the current word in it

   B) Reverse temp

   C) Append temp to st with a blank space

6) Print st

32.2 Code

```
import java.util.Scanner;
public class ReverseEachWord
{
   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter String");
      String inp = s.nextLine();    //String to store input

      String[] words = inp.split(" "); //Splits string at each space (each word)
      StringBuffer st = new StringBuffer(""); //New empty StringBuffer to store reverse of
          string

      for(int i = 0; i < words.length; i++) //Loop to iterate through array formed by splitting
          string
      {
         StringBuffer temp = new StringBuffer(words[i]); //StringBuffer to store each word
         temp.reverse();                       //Each word is reversed
         st.append(temp + " ");                //Reversed word is appended to StringBuffer st
      }

      System.out.println(st);

      s.close();
   }
}
```

33 Checking Increasing, Decreasing and Bouncy Numbers

Write a program to check if a given number is Increasing, Decreasing or Bouncy.
Increasing Number : Working from left-to-right if no digit is exceeded by the digit to its left it is called an increasing number; for example, 22344.
Decreasing Number : Similarly if no digit is exceeded by the digit to its right it is called a decreasing number; for example, 774410.
Bouncy Number : We shall call a positive integer that is neither increasing nor decreasing a bouncy number; for example, 155349. Clearly there cannot be any bouncy numbers below 100.

Input:
123456

Output:
Increasing

33.1 Algorithm

1) Declare a Scanner object to take user input

2) Take user input of a number and store it in an int (n)

3) Check if the number is Increasing using isIncreasing(int n) and display message if it is

4) Check if the number is Decreasing using isDecreasing(int n) and display message if it is

5) If the number is neither Increasing or Decreasing, it is Bouncy. Display appropriate message.

boolean isIncreasing(int n):

1) Parse the number to a String(st)

2) Declare a boolean(inc) to flag if it is Increasing and set to true

3) Run a loop from i = 0 to length of String - 1, incrementing i at each iteration

   A) if the character at i $\geq$ character preceding it, continue the loop

   B) else, set inc to false

4) Return inc

boolean isDecreasing(int n):

1) Parse the number to a String(st)

2) Declare a boolean(dec) to flag if it is Decreasing and set to true

3) Run a loop from i = 0 to length of String - 1, incrementing i at each iteration

   A) if the character at i $\leq$ character preceding it, continue the loop

   B) else, set dec to false

4) Return dec

33.2 Code

```
import java.util.Scanner;
public class Bouncy_Inc_DecCheck
{
   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter Number to Check");
      int n = s.nextInt();              //Variable to store number

      if(isIncreasing(n))                  //Checks if number is increasing
         System.out.println("Increasing");
      else if(isDecreasing(n))             //Checks if number is decreasing
         System.out.println("Decreasing");
      else
         System.out.println("Bouncy");     //Else number is bouncy Number

      s.close();
   }


   public static boolean isIncreasing(int n)  //Function to check if number is increasing
   {
      String st = Integer.toString(n);   //Converts integer to String

      boolean inc = true;                //Flag to indicate Increasing Number

      for(int i = 1; i < st.length(); i++)  //Loop to iterate through string
      {
         if(st.charAt(i) >= st.charAt(i - 1)) //Checks if character is more than or equal to
             preceding number
            continue;
         else
            inc = false;
      }

      return inc;  //returns boolean flag
   }

   public static boolean isDecreasing(int n)  //Function to check if number is decreasing
   {
      String st = Integer.toString(n);   //Converts integer to String

      boolean dec = true;                //Flag to indicate Increasing Number

      for(int i = 1; i < st.length(); i++)  //Loop to iterate through string
      {
         if(st.charAt(i) <= st.charAt(i - 1)) //Checks if character is more than or equal to
             preceding number
            continue;
         else
            dec = false;
      }

      return dec;  //returns boolean flag
   }
}
```

34 String Negative Encoder

Write a program to encode a String using negative encoding of characters with cyclic encoding with a given value of encoding.

Input:
Java Concept Of The Day
Value = 2

Output:
HYTY AMLACNR MD RFC BYW

34.1 Algorithm

1) Declare a Scanner object tot ake user input

2) Take user input of a String to encode(input) and convert it to UpperCase

3) Take input for an encoding value(n)

4) Declare a StringBuffer(str) and set it to input

5) Run a loop from i = 0 to length of String - 1, incrementing i at each iteration

   A) If the character is a blank space, skip it

   B) Store the int value of the current character in an int(x)

   C) Subtract the encoding value from x

   D) if x < 65 (Before 'A')

      i) set x to 64−x

      ii) Replace the character with character having the value 'Z' − x

   E) else

      i) Replace the character with character having the value of character − n

6) Print the new String

34.2 Code

```
import java.util.Scanner;

public class String_NegativeEncoder
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter Word to Encode");
      String input = s.nextLine().toUpperCase();  //Convert input to UpperCase

      System.out.println("Enter Encoding value:");
      int n = s.nextInt();               //Value to encode by

      StringBuffer str = new StringBuffer(input.trim()); //To perform edits

      int x, i;  //Temporary variables for operations
```

```
      for(i = 0; i < str.length(); i++)
      {
         if(str.charAt(i) == ' ') //Skips if it is a space
            continue;

         x = (int)str.charAt(i); //Storing int value of a character
         x = x - n; //Subtracts encoding value

         if(x < 65) //Cyclic shift for characters before 'A'
         {
            x = 64 - x; //Converts it to x characters before 'X'
            str.setCharAt(i, (char)((int)'Z'- x)); //Replaces character in String
         }
         else
         {
            str.setCharAt(i, (char)((int)str.charAt(i) - n)); //Replaces character in String
         }
      }

      System.out.println("Encoded Word: " + str.toString()); //Printing
   }

}
```

35 Calculate Number of Days Past In The Year Till a Given Date

Write a program to calculate the number of days past in a year before a given date, checking for all appropriate conditions.

Input:
29 05 1999

Output:
149

35.1 Algorithm

1) Declare a Scanner object to take user input

2) Take user input of the date

3) Declare an int(days) to store total number of days

4) Check validity of input, with conditions to check leap year and the dates accordingly

5) Add a switch block, adding the days of all preceding months using a fall-through

6) Add the date to days(number of days in that month)

7) Print days

35.2 Code

```
import java.util.Scanner;

public class NoOfDays
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter date in dd mm yyyy format.");
      int d = s.nextInt(); //Date
      int m = s.nextInt(); //Month
      int y = s.nextInt(); //Year
      int days = 0; //Number of days

      if(d <= 0 || m <= 0 || y <= 0 || d > 31 ) //Checks if input is within bounds
      {
         System.out.println("Invalid Date");
         System.exit(0);
      }

      boolean leap = isLeapYear(y); //Checks if it is a leap year
      if((leap && d > 29 && m == 2) || (!leap && d > 28 && m == 2)) //Checks entered date in
          case of February
      {
         System.out.println("Invalid Date");
         System.exit(0);
      }
```

```
      if(m == 4 || m == 6 || m == 9 || m == 11) //Checks entered date in case of months with 30
          days
      {
         if(d > 30)
         {
            System.out.println("Invalid Date");
            System.exit(0);
         }
      }

      switch(m)  //Adds days for all months before the entered one
      {
         case 12: days += 30;

         case 11: days += 31;

         case 10: days += 30;

         case 9:    days += 31;

         case 8:    days += 31;

         case 7:    days += 30;

         case 6:    days += 31;

         case 5:    days += 30;

         case 4:    days += 31;

         case 3:    if(leap)
                    days += 29;
                 else
                    days += 28;

         case 2:    days += 31;
      }

      days += d; //Days in the entered month

      System.out.println(days);
   }

   public static boolean isLeapYear(int year) //Checks for leap year
   {
      return (year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0));
   }

}
```

36 Wondrous Square and Prime Display

Write a program to check whether a matrix is a Wondrous Square or not and display each prime and its index.

A wondrous square is an n×n grid fulfilling the conditions:

1) Contains integers from 1 to n^2

2) Sum of integers in any row or column adds up to $0.5 * n * (n^2 + 1)$

Input:
Order = 5

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

Output:
Wondrous Square

Prime	Row	Column
17	0	0
23	1	0
5	1	1
7	1	2
13	2	2
19	3	2
3	3	4
11	4	0
2	4	3

36.1 Algorithm

1) Declare a Scanner object to take user input

2) Take user input for the order of the matrix and declare an appropriate 2-D int matrix(A[ ][ ])

3) Run a loop from i = 0 to n - 1, incrementing i at each iteration

   A) Run a loop from j = 0 to n - 1, incrementing j at each iteration

      i) Take user input and check if it is less than 0 or more than n^2

      ii) Add element to matrix at A[i][j]

4) Print the Matrix

5) Declare a boolean(flag) to riase a flag if sum does not match condition, and an int(sum) to store the sum according to condition

6) Run a loop from i = 0 to n - 1, incrementing i at each iteration

   A) Run a loop from j = 0 to n - 1, incrementing j at each iteration

      i) Add A[i][j] and A[j][i] in two separate variables (Summing up each row and column)

   B) Check if both sums match the defined value(sum) and continue if true, raising a flag if not

7) Check for flag and display appropriate message

8) Run a loop to iterate through the array and check each element for prime, displaying it if it is

boolean checkPrime(int n):

1) if n is 0 or 1, return false

2) if n is 2, return true

3) if n is a multiple of 2, return false

4) Run a loop from 3 to sqrt(n) and increment by 2 at each iteration

A) If n is divisible by the current index, return false, else continue loop

5) return true

36.2 Code

```
import java.util.Scanner;

public class WondrousSquare
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      System.out.println("Enter Order of Matrix.");
      int n = s.nextInt();

      int i, j, k, A[][] = new int[n][n];

      System.out.println("Enter Elements of Matrix Row-Wise");

      for(i = 0; i < n; i++)
         for(j = 0; j < n; j++)
         {
            int x = s.nextInt();
            if(x < 0 || x > n*n)
            {
               System.out.println("Invalid Input");
               System.exit(0);
            }
            A[i][j] = x;
         }

      System.out.println();

      for(i = 0; i < n; i++)
      {
         for(j = 0; j < n; j++)
            System.out.print(A[i][j] + "\t");
         System.out.println();
      }


      boolean flag = false;
      double sum = 0.5 * n * (n * n + 1);

      for(i = 0; i < n; i++)
      {
         double sumc = 0, sumr = 0;
         for(j = 0; j < n; j++)
```

```
        {
           sumc += A[i][j];
           sumr += A[j][i];
        }


        if(sumc != sum || sumr != sum)
        {
           flag = true;
           break;
        }
     }

     if(flag)
        System.out.println("Not Wondrous Square");
     else
        System.out.println("Wondrous Square");

     System.out.println("\n\nPrime\tRow\tColumn");
     for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
           if(checkPrime(A[i][j]))
              System.out.println(A[i][j] + "\t" + i + "\t" + j);
  }

  public static boolean checkPrime(int n)
  {
     if(n == 1)
        return false;
     if(n == 2)
        return true;
     if (n % 2 == 0)
        return false;
      for(int i = 3; i * i <= n; i += 2)
          if(n % i == 0)
              return false;

      return true;
  }

}
```

37 Matrix Multiplication

Write a Program to multiply two matrices.

Input:
Matrix 1:
1 3 5
6 7 8
9 4 5

Matrix 2:
2 12 6
8 9 7
3 4 5

Output:
Multiplied Matrix:
41 59 52
92 167 125
65 164 107

37.1 Algorithm

1) Declare a Scanner object to take user input

2) Take user input for dimensions of the two matrices and check if columns in first = rows in second

3) Create two arrays (A[ ][ ] and B[ ][ ]) of appropriate size

4) Take user input for the elements of the matrices

5) Print the original matrices

6) Create an array (C[ ][ ]) for the multiplication of the size Rows1 × Columns2

7) Run a loop from i = 0 till Rows1 - 1, incrementing i at each iteration

   A) Run a loop from j = 0 till Columns2 - 1, incrementing j at each iteration

      i) Run a loop from k = 0 till Rows1 - 1, incrementing k at each iteration

         a) Add A[i][k] × B[k][j] to C[i][j] (Row of first matrix with column of second)

8) Print the Multiplied Matrix

37.2 Code

```
import java.util.Scanner;

public class Multiplication
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      int r1, r2, c1, c2, i, j;

      //Dimensions of the two Matrices
      System.out.println("Enter Number of Rows in First Matrix:");
      r1 = s.nextInt();

      System.out.println("Enter Number of Columns in First Matrix:");
```

```
c1 = s.nextInt();

System.out.println("Enter Number of Rows in Second Matrix:");
r2 = s.nextInt();

System.out.println("Enter Number of Columns in Second Matrix:");
c2 = s.nextInt();


if(c1 != r2)  //Condition for multiplication
{
   System.out.println("Number of Columns in Matrix 1 have to be equal to Rows in Matrix
       2");
   System.exit(0);
}

//Creating the Matrices
int A[][] = new int[r1][c1];
int B[][] = new int [r2][c2];

//Data Entry
System.out.println("Enter the elements in the First Matrix, Row-Wise:");

for(i = 0; i < r1; i++)
   for(j = 0; j < c1; j++)
      A[i][j] = s.nextInt();


System.out.println("Enter the elements in the Second Matrix, Row-Wise:");

for(i = 0; i < r2; i++)
   for(j = 0; j < c2; j++)
      B[i][j] = s.nextInt();

System.out.println("\n\n\n");

//Printing the Original Matrices
System.out.println("Matrix 1:");

for(i = 0; i < r1; i++)
{
   for(j = 0; j < c1; j++)
      System.out.print(A[i][j] + "\t");
   System.out.println();
}

System.out.println("Matrix 2:");

for(i = 0; i < r2; i++)
{
   for(j = 0; j < c2; j++)
      System.out.print(B[i][j] + "\t");
   System.out.println();
}

int C[][] = new int [r1][c2]; //Multiplication array

for( i = 0; i < r1; i++)
{
   for( j = 0;j < c2; j++)
   {
```

```
            for(int k = 0;k < r1; k++)
            {
                  C[i][j] += A[i][k] * B[k][j]; //Multiplies Row of 1st matrix with Column of
                      the other and adds to third
              }
           }
      }

      System.out.println("Multiplied Matrix:"); //Printing the Multiplied Matrix

      for(i = 0; i < r1; i++)
      {
         for(j = 0; j < c2; j++)
            System.out.print(C[i][j] + "\t");
         System.out.println();
      }

      s.close();
   }

}
```

38 Towers Of Hanoi

Write a program to solve the Towers of Hanoi problem for a given N number of disks and 3 pegs.
The object is to move all disks from peg A to C, one at a time such that a larger disk is never over a smaller one.

Input:
N = 5

Output:
(Pegs Labelled as L - C - R)
Move disk 1 from L to R
Move disk 2 from L to C
Move disk 1 from R to C
Move disk 3 from L to R
Move disk 1 from C to L
Move disk 2 from C to R
Move disk 1 from L to R
Move disk 4 from L to C
Move disk 1 from R to C
Move disk 2 from R to L
Move disk 1 from C to L
Move disk 3 from R to C
Move disk 1 from L to R
Move disk 2 from L to C
Move disk 1 from R to C
Move disk 5 from L to R
Move disk 1 from C to L
Move disk 2 from C to R
Move disk 1 from L to R
Move disk 3 from C to L
Move disk 1 from R to C
Move disk 2 from R to L
Move disk 1 from C to L
Move disk 4 from C to R
Move disk 1 from L to R
Move disk 2 from L to C
Move disk 1 from R to C
Move disk 3 from L to R
Move disk 1 from C to L
Move disk 2 from C to R
Move disk 1 from L to R

38.1 Algorithm

1) Take input for number of disks and store it in an int(N)

2) Define the pegs with characters

3) Call move(), passing N, start, intermediate and destination peg characters as parameters

void move(int N, char startPeg, char intPeg, char destPeg):

1) If $N \neq 0$

A) Move N - 1 disks from startPeg to intPeg (call move with parameters as N, startPeg, destPeg, intPeg)

B) Move disk N from startPeg to destPeg

C) Move N - 1 disks from intPeg to destPeg (call move with parameters as N, intPeg, startPeg, destPeg)

38.2 Code

```
import java.util.Scanner;

public class TowersOfHanoi
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      char start = 'L', inter = 'C', dest = 'R';

      System.out.println("Enter number of disks");
      int N = s.nextInt();  //Number of disks
      TowersOfHanoi t = new TowersOfHanoi();
      t.move(N, start, inter, dest);
   }

   private void move(int N, char startPeg, char intPeg, char destPeg)
   {
      if(N != 0)
      {
         //Move n - 1 disks from starting peg to intermediate peg
         move(N-1, startPeg, destPeg, intPeg);
         //Move disk N from start to Destination
         System.out.println("Move disk " + N + " from " + startPeg + " to " + destPeg);
         //Move n - 1 disks from intermediate peg to destination peg
         move(N-1, intPeg, startPeg, destPeg);
      }
   }

}
```

39 Unique 2 Digit Combinations

Write a program that asks the user to enter an integer and print all unique two digit integers that can be formed using the digits of that number

Input:
1231

Output:
11 12 13 21 23 31 32

39.1 Algorithm

1) Take user input of a number(num) and parse each digit to the elements of an array(digits[ ]) and store its length

2) Create a new StringBuffer(str)

3) Run a loop from i = 0 till length - 1, incrementing i at each iteration

   A) Append digits[0] and digits[i] with a blankspace to str

4) Run a loop from i = 1 to length - 2, incrementing i at each iteration

   A) Run a loop from j = i - 1 till j $\geq$ 0, decrementing j at each iteration

   B) Append digits[i] and digits[j] with a blank space to str

   C) Run a loop from j = i + 1 till j $\leq$ length, incrementing j at each iteration

   D) Append digits[i] and digits[j] with a blank space to str

5) Run a loop from i = length - 2 till i $\geq$ 0, decrementing i at each iteration

6) Split the StringBuffer into a String array from blank spaces

7) Parse the Strings into an int array(numbers[ ])

8) Sort numbers[ ] in ascending order

9) Create anew int array(uniques[ ]) to store unique elements

10) Compare elements in unique with numbers, adding all unique elements

11) Print unique elements

39.2 Code

```
import java.util.Scanner;

public class Unique2DigitCombos
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      int i, j, temp;

      System.out.println("Enter a number");
      int num = s.nextInt();

      int length = String.valueOf(num).length(); //Length of input
```

```
int[] digits = new int[length];  //Array of digits
for(i = 0; i < length; i++) //Parses each digit to an elemnt in array
{
   digits[i] = num % 10;
   num = num / 10;
}

StringBuffer str = new StringBuffer(""); //For editing

for(i = 0; i < length; i++)
   str.append(digits[0] + "" + digits[i] + " "); //Appends all digits with last one

for(i = 1; i < length - 1; i++)
{
   for(j = i - 1; j >= 0; j--) //iterates though digits and appends with all before them
      str.append(digits[i] + "" + digits[j] + " ");
   for(j = i + 1; j <= length - 1; j++) //iterates though digits and appends with all
       after them
      str.append(digits[i] + "" + digits[j] + " ");
}

for(i = length - 2; i >= 0; i--)
   str.append(digits[length - 1] + "" + digits[i] + " "); //Appends first digit with all
       before it

String[] num_str = str.toString().split(" "); //Splits StringBuffer into numbers
int numbers[] = new int[num_str.length]; //Array for numbers


for(i = 0; i < numbers.length;i++)
   numbers[i] = Integer.parseInt(num_str[i]); //Parses each String to int

for(i = 0; i < numbers.length; i++)  //Sorts in ascending order
{
   for(j = 0; j < numbers.length - 1; j++)
   {
      if(numbers[j + 1] < numbers [j]) //Change this to "Array[j + 1] > Array [j]" to sort
          in descending order
      {
         temp = numbers[j + 1];
         numbers[j + 1] = numbers [j];
         numbers[j] = temp;
      }
   }
}


int[] uniques = new int[numbers.length]; //Array of unique numbers

for(i = 0; i < numbers.length; i++) //initializes all elements to 0
{
   uniques[i] = 0;
}
uniques[0] = numbers[0];

for(i = 1, j = 1; i < numbers.length && j < uniques.length - 1; i++)
{
   if(numbers[i] == uniques[j]) //Compares consecutive number with element in uniques,
       skips if equal
      continue;
   else
```

```
        {
           j++; //Go to next element
           uniques[j] = numbers[i]; //Set uniques to number
        }
     }

     for(i = 0; i < uniques.length; i++)
        if(uniques[i] == 0)
           continue;
        else
           System.out.print(uniques[i] + " ");

  }

}
```

40 2-D Array Sort

Write a program to sort a 2-D array in ascending order

Input:
5 4 2
3 5 4
8 9 1

Output:
1 2 3
4 4 5
5 8 9

40.1 Algorithm

1) Take user input for Dimensions fo the matrix.

2) Create a 2D array(A[ ][ ]) of given dimensions and a normal array(B[ ]) of size product of dimensions

3) Take user input for data, entering data in both arrays simultaneously

4) Sort B in ascending order

5) Add each element from B to A in order

6) Print A

40.2 Code

```
import java.util.Scanner;

public class ArraySort2D
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter Order of Matrix:");
      int n = s.nextInt(); //Order of the Matrix

      int A[][] = new int[n][n], B[] = new int[n*n], k = 0, i, j; //A is original Matrix, B is
          for sorting

      System.out.println("Enter elements of array Row-wise:");
      //Data Entry
      for(i = 0; i < n; i++)
         for(j = 0; j < n; j++)
         {
            int x = s.nextInt();
            if(x < 0)
            {
               System.out.println("Invalid Input. Input cannot be less than 0");
               System.exit(0);
            }
            A[i][j] = x;  //Adds to A
            B[k] = x;  //Adds to B
            k++;
```

```
      }

   for(i = 0; i < n * n; i++) //Sorting B
      for(j = 0; j < n * n - 1; j++)
         if(B[j + 1] < B[j])
         {
            int temp = B[j];
            B[j] = B[j + 1];
            B[j + 1] = temp;
         }

   k = 0;
   for(i = 0; i < n; i++) //Copies elements from B to A in order
      for(j = 0; j < n; j++, k++)
         A[i][j] = B[k];

   for(i = 0; i < n; i++) //Printing
   {
      for(j = 0; j < n; j++)
      {
         System.out.print(A[i][j] + "\t");
      }
      System.out.println();
   }

  }

}
```

41 Mirror Image Of A Matrix

Write a program to find the mirror image of a matrix

Input:
1 2 5
3 4 5
8 7 9

Output:
5 2 1
5 4 3
9 7 8

41.1 Algorithm

1) Take user input for dimensions of the matrix

2) Declare two arrays(A[ ][ ] and B[ ][ ]) of appropriate size

3) Run a loop from i = 0 to rows - 1, incrementing i at each iteration

A) Run a loop from j = 0 and k = columns - 1 till j < n and k ≥ 0, incrementing j and decrementing j by 1 at each iteration

i) Take user input

ii) Add it to A[i][j]

iii) Add it to B[i][k] (Inverted column)

4) Print Matrix A and B

41.2 Code

```
import java.util.Scanner;

public class MirrorImage
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      int m,n, i, j, k, temp;
      //Dimensions of the Matrix
      System.out.println("Enter Number of Rows:");
      m = s.nextInt();

      System.out.println("Enter Number of Columns:");
      n = s.nextInt();

      int A[][] = new int[m][n], B[][] = new int[m][n]; //A is original, B is mirror

      System.out.println("Enter the elements in the array, Row-Wise:"); //Data Entry

      for(i = 0; i < m; i++)
         for(j = 0, k = n - 1; j < n && k >= 0; j++, k--)
         {
            temp = s.nextInt();
            A[i][j] = temp; //Add to A normally
```

```
            B[i][k] = temp; //Add to B with columns in opposite order
         }

      System.out.println("Your array:");
      //Printing
      for(i = 0; i < m; i++)
      {
         for(j = 0; j < n; j++)
            System.out.print(A[i][j] + "\t");
         System.out.println();
      }

      System.out.println("Mirror array:");

      for(i = 0; i < m; i++)
      {
         for(j = 0; j < n; j++)
            System.out.print(B[i][j] + "\t");
         System.out.println();
      }

   }

}
```

42 Rotate Matrix by 90 Degrees Clockwise

Input:
1 2 3
4 5 6
7 8 9

Output:
7 4 1
8 5 2
9 6 3

42.1 Algorithm

1) Take user input for order of matrix

2) Declare an array(A[ ][ ]) of appropriate size

3) Take user input for data

4) Print the original matrix

5) Run a loop from i = 0 to n - 1, incrementing i at each iteration

A) Run a loop from j = n - 1 till j $\geq$ 0, decrementing j at each iteration

i) Print A[j][i]

42.2 Code

```
import java.util.Scanner;
public class MatrixRotation90
{

   public static void main(String[] args)
   {

      Scanner s = new Scanner(System.in);
      int m,n, i, j;
      //Dimensions
      System.out.println("Enter Order of Matrix:");
      n = s.nextInt();

      int A[][] = new int[n][n]; //Matrix

      System.out.println("Enter the elements in the array, Row-Wise:");
      //Data Entry
      for(i = 0; i < n; i++)
         for(j = 0; j < n; j++)
            A[i][j] = s.nextInt();

      System.out.println("Your Matrix:");
      //Original Matrix
      for(i = 0; i < n; i++)
      {
         for(j = 0; j < n; j++)
            System.out.print(A[i][j] + "\t");
         System.out.println();
      }
```

```
      System.out.println("Rotated Matrix:");
      for(i = 0; i < n; i++)
      {
         for(j = n - 1; j >= 0; j--) //Transposes and prints rows in opposite direction
            System.out.print(A[j][i] + "\t");
         System.out.println();
      }

      s.close();
   }

}
```

43 Insertion Sort (Array)

Write a program to implement Insertion Sort in an array

43.1 Algorithm

1) Create an array and take user input of data to be sorted

2) Run a loop from i = 0 to length of array - 1, incrementing i by 1 at each iteration

   A) Store A[i] in an int(x)

   B) Run a loop from j = i - 1 till j $\geq$ 0 and A[j]x, decrementing j by 1 at each iteration

      i) Assign A[j] to A[j + 1] (Shifts elements right to make place for x)

   C) Assign x to A[j + 1]

43.2 Code

```
import java.util.Scanner;
public class InsertionSortArray
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);
      int A[] = new int[10];
      int i, x, j;

      System.out.println("Enter 10 elements:");
      for(i = 0; i < 10; i++) //Data Entry
         A[i] = s.nextInt();

      for(i = 0; i < 10; i++)
      {
         x = A[i];  //Takes current element

         for(j = i - 1; j >= 0 && A[j] > x; j--) //Shifts all elements ahead one space to make
             place for sorting element
            A[j + 1] = A[j];

         A[j + 1] = x; //Places element in space
      }

      for(i = 0; i < 10; i++) //Printing
         System.out.print(A[i] + " ");

   }

}
```

44 Binary Search (Recursive)

Write a program to implement recursive binary search in an array.

44.1 Algorithm

1) Take user input of an array or use a sorted sample array (sample used here)

2) Take user input for data to search for

3) Call binarySearch(), passing the array, element, 0 and length - 1 as parameters

4) Print position if function does not return -1, else display appropriate message

int binarySearch(int[] A, intx, int low, int high)

1) If low > high (out of bounds), return -1 (not found)

2) Set mid to (low + high) / 2

3) If A[mid] < x, call binarySearch, passing A, x, mid + 1 and high as parameters and return its value

4) If A[mid] > x, call binarySearch, passing A, x, low and mid - 1 as parameters and return its value

44.2 Code

```
import java.util.Scanner;
public class RecursiveBinarySearch
{
   public static void main(String[] args)
   {
      int A[] = {2, 45, 69, 234, 567, 876, 900, 976, 999}; //Sample Data
      Scanner s = new Scanner(System.in);

      System.out.println("Enter data to search for:");
      int x = s.nextInt();

      int found = binarySearch(A, x, 0, A.length - 1);

      if(found != -1)
         System.out.println("Found at position " + (found + 1));
      else
         System.out.println("Not Found");
   }

   static int binarySearch(int A[], int x, int low, int high)
   {
      if(low > high)
         return -1;
      int mid = (low + high) / 2;
      if(x > A[mid])
         return binarySearch(A, x, mid + 1, high);
      else if(x < A[mid])
         return binarySearch(A, x, low, mid - 1);
      return mid;
   }
}
```

45 Decimal To Roman Numerals

Write a Program to Convert decimal numbers to their Roman equivalent

Input:
969

Output:
CMLXIX

45.1 Algorithm

1) Declare a String array(Hundreds[ ]) and fill it wih Hundreds in Roman Numerals(C, CC ... CM)

2) Declare a String array(Tens[ ]) and fill it with Tens in Roman Numerals(X, XX ... XC)

3) Declare a String array(Units[ ]) and fill it with Units in Roman Numerals(I, II ... IX)

4) Take user input of number

5) Hundreds is found by taking the element at n / 100 in Hundreds[ ]

6) Tens is found by taking element at (n / 10) % 10 in Tens[ ]

7) Units is found by taking element at n % 10 in Units[ ]

45.2 Code

```
import java.util.Scanner;
public class DecimalRoman
{

   public static void main(String[] args)
   {
      Scanner s = new Scanner(System.in);

      String[] Hundreds = {"", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"}; //Array
          for Hundreds in Roman
      String Tens[]={"", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"}; //Array for
          Tens in Roman
      String Units[]={"", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"}; //Array for
          Units in Roman

      System.out.println("Enter Decimal Number to Convert to Roman (Less than 1000):"); //Input
      int n = s.nextInt();

      String Hund = Hundreds[n / 100]; //Division by 100 yields number of hundreds
      String Ten = Tens[(n / 10) % 10]; //Division by 10 and the remainder of further division
          by 10 yields Tens
      String Unit = Units[n % 10]; //Remainder of division by 10 yields number of Ones

      System.out.println("Roman Equivalent: " + Hund + Ten + Unit); //Final Answer
   }

}
```

46 Linked List

Write a program to create and use a LinkedList (Node class used directly from the linkedlist package, included in Other Resources)

46.1 Algorithm

1) Declare global Nodes(start and end), and an int(size)

2) Run LLRun() for a menu-driven use fo Linked List

3) Display choices for Insert at Start, End and Position, Delete from Start, End and Position and for Display and Reversal of the List

4) Run appropriate functions based on the user choice and take input of required data before call

void insertAtStart(int val):

1) Create a new Node with the given value and a null Link

2) Increment size by 1

3) If List is empty,

   A) Set start and end to the new Node

4) Else

   A) Set link of the new Node to start

   B) Set start to the new Node

void insertAtEnd(int val):

1) Create a new Node with the given value and link as null

2) Increment size by 1

3) If List is empty, set start and end to the new Node

4) Else

   A) Set link of end to the new Node

   B) Set end to the new Node

void insertAtPosition(int index, int val):

1) Check if the input is within range

2) If index = 1, call insertAtStart, if index = size + 1, call insertAtEnd

3) Create a new Node with the given value and link as null

4) Else,

   A) Run a loop from i = 0 to index - 1, with a temporary Node(ptr) starting from start, incrementing i at each iteration and setting ptr to the next Node in List at each iteration

   B) If i = index - 1

      i) Set link of the new Node to ptr.link

ii) Set link of ptr to the new Node

iii) Increment size of the List

void deleteFromStart():

1) If List is empty, Print appropriate message
2) Else
   A) Decrement Size by 1
   B) Set start to start.link

void deleteFromEnd():

1) If List is empty, print appropriate message
2) If list has 1 element
   A) Print data in Start
   B) Set start and end to null
   C) Set size to zero
3) Else
   A) Create a new Node(ptr) and iterate to second last element in the List
   B) Print data in ptr.link
   C) Set end to ptr
   D) Decrement size by 1

void deleteFromIndex(int index):

1) If index = 1, call deleteFromStart and if index = size, call deleteFromEnd
2) Else
   A) Create two Nodes(temp and ptr) and set them to start
   B) Iterate ptr to the Node before Node at given index
   C) Iterate temp to the second Node after ptr
   D) Print data in ptr.link
   E) Set link of ptr to temp
   F) Decrement size by 1

void display():

1) If Size = 0, display appropriate message
2) Else
   A) Create a new Node(pointer) and set it equal to start of the List
   B) Run a loop from i = 0 to size - 1, incrementing i at each iteration
      i) Print data in pointer

ii) Set pointer to the next Node in the List

void reverse():

1) Run a loop from i = 1 till i $\leq$ size / 2, incrementing i at each iteration

A) Declare a temporary int(temp)

B) Create two Nodes(ptr1 and ptr2) and set them to start

C) Iterate ptr1 to the node at the 'i'th index and ptr2 to (size - i)th index

D) Swap data in both Nodes

46.2 Code

```
package linkedList;
import java.util.Scanner;

public class LinkedList
{
   public Node start; //First Node
   public Node end; //Last Node
   public int size; //Size of LinkedList

   public LinkedList() //Default Constructor
   {
      start = end = null;
      size = 0;
   }

   public LinkedList(int[] a) //To initialize LinkedList with given data
   {
      for(int i = 0; i < a.length; i++) //Adds each data element to List
         insertAtEnd(a[i]);
   }

   public LinkedList(Node start, int nodes) //Adds data from new LinkedList to Current one,
       given the number of nodes in the other List
   {
      for(int i = 0; i < nodes; i++)
      {
         insertAtEnd(start.getData()); //Inserts First node of new List at the end of the
             current one
         start = start.getLink(); //Sets to next Node in list
      }
   }

   public boolean isEmpty() //To check if List is empty
   {
      return start == null;
   }

   public int getSize() //Returns size of List
   {
      return size;
   }

   public void insertAtStart(int val) //Function to insert Node at start of the List
```

```
{
   Node nptr = new Node(val, null); //Node to be inserted
   size++;

   if(start == null) //For empty list
   {
      start = nptr;
      end = start;
   }
   else //Non-empty List
   {
      nptr.setLink(start);
      start = nptr;
   }
}

public void insertAtEnd(int val) //Function to insert Node at the end of the List
{
   Node nptr = new Node(val, null); //Node to be inserted
   size++;

   if(start == null) //Empty List
   {
      start = nptr;
      end = start;
   }
   else
   {
      end.setLink(nptr);
      end = nptr;
   }
}

public void insertAtIndex(int index, int val) //Function to insert Node at a given index
{
   if(index > size + 1 || index < 1) //Checks if index is in range of List
   {
      System.out.println("Index does not exist");
      return;
   }
   else if(index == 1)
      insertAtStart(val);
   else if(index == size + 1)
      insertAtEnd(val);
   else
   {
      Node nptr = new Node(val, null);
      Node ptr = start;

      for(int i = 1; i < size; i++, ptr = ptr.getLink()) //To find Node before index to
           insert at
         if(i == index - 1)
         {
            nptr.setLink(ptr.getLink());
            ptr.setLink(nptr);
            size++;
            break;
         }
   }
}
```

```
public void deleteFromStart() //Function to delete Node from the front
{
   if(size == 0)
   {
      System.out.println("List Empty");
      return;
   }

   start = start.getLink();
   size--;
}

public void deleteFromEnd()   //Function to delete Node from the end
{
   if(size == 0)
   {
      System.out.println("List Empty");
      return;
   }
   if(size == 1)
   {
      System.out.println(start.getData());
      start = end = null;
      size = 0;
      return;
   }
   Node ptr = start;

   for(; ptr.getLink() != end;) //Iterates through list to Node before last Node
      ptr = ptr.getLink();
   System.out.println(ptr.getLink().getData());
   end = ptr; //Sets last Node as end
   end.setLink(null);
   size--;
}

public void deleteFromIndex(int index) //Function to delete Node at a certain Index
{
   if(index == 1)
      deleteFromStart();
   if(index == size)
      deleteFromEnd();
   else
   {
      Node ptr = start;
      Node temp = start;
         for(int i = 1; i < size; ptr = ptr.getLink()) //Iterates to Node before required Node
            if(i == index - 1)
            {
               temp = ptr.getLink();
               temp = temp.getLink();
               ptr.setLink(temp);
               size--;
               break;
            }
   }
}

public void display() //Function to print the List
{
   if(size == 0)
```

```
   {
      System.out.println("Empty List");
      return;
   }
   else
   {
      System.out.println("\n\n\n");
      Node pointer = start;
      for(int i = 0; i < size; i++)
      {
         System.out.print(pointer.getData() + "\t");
         pointer = pointer.getLink();
      }
      System.out.println();
   }
}


public int inpData() //To take input of data for Node
{
   Scanner s = new Scanner(System.in);
   System.out.println("Enter data for Node");
   int x = s.nextInt();
   s.close();
   return x;
}

public void reverse() //Function to reverse the List by swapping
{
   for(int i = 1; i <= size / 2; i++)
   {
      int temp;
      Node ptr1 = start;
      Node ptr2 = start;
      for(int j = 1; j < i; j++)
         ptr1 = ptr1.getLink();
      for(int k = 1; k <= size - i; k++)
         ptr2 = ptr2.getLink();

      temp = ptr1.getData();
      ptr1.setData(ptr2.getData());
      ptr2.setData(temp);
   }
}

public void LLRun() //Function for menu-driven Usage
{
   int choice, val, pos;
   Scanner s = new Scanner(System.in);

   do
   {
      System.out.println("\n\n");
      System.out.println("1 - Insert At Beginning");
      System.out.println("2 - Insert At End");
      System.out.println("3 - Insert At Position");
      System.out.println("4 - Delete From Beginning");
      System.out.println("5 - Delete From End");
      System.out.println("6 - Delete From Position");
      System.out.println("7 - Display List");
      System.out.println("8 - Return to Linked List type selection");
```

```
      System.out.println("10 - Reverse Linked List");
      System.out.println("9 - Exit");
      choice = s.nextInt();

      switch(choice)
      {
         case 1:    System.out.println("\nEnter value");
                    val = s.nextInt();
                    insertAtStart(val);
                    break;

         case 2:    System.out.println("\nEnter value");
                    val = s.nextInt();
                    insertAtEnd(val);
                    break;

         case 3:    System.out.println("\nEnter Position and value");
                    pos = s.nextInt();
                    val = s.nextInt();
                    insertAtIndex(pos, val);
                    break;

         case 4:    deleteFromStart();
                    break;

         case 5:    deleteFromEnd();
                    break;

         case 6:    System.out.println("\nEnter Position");
                    pos = s.nextInt();
                    deleteFromIndex(pos);
                    break;


         case 7:    display();
                    break;

         case 8:    return;

         case 10: reverse();
                    break;

         case 9:    System.exit(0);



         default: System.out.println("\nEnter valid Choice");
                    break;
      }
   }while(choice != 9);
   s.close();
}


public static void main(String args[]) //Menu to choose between Linked List, Stack or Queue
    (All usind LinkedLists)
{
   int choice = 0;
   Scanner s = new Scanner(System.in);
   LinkedList l = new LinkedList();
   Stack st = new Stack();
```

```
      Queue q = new Queue();

      do
      {
         System.out.println("1 - Linked List");
         System.out.println("2 - Stack");
         System.out.println("3 - Queue");
         System.out.println("9 - Exit");
         choice = s.nextInt();

         switch(choice)
         {
            case 1:    l.LLRun();
                    break;

            case 2:    st.StackRun();
                    break;

            case 3:    q.QueueRun();
                    break;

            case 9:    System.exit(0);

            default: System.out.println("\nEnter valid Choice");
         }
      }while(choice != 9);
      s.close();
   }
}
```

47 Queue (Linked List)

Write a Program to implement Queue using Linked List (Node class used directly, can be found in Other Resources)

47.1 Algorithm

1) Display Menu for user to select operation

2) For Insertion, take input of data and call insertAtEnd(), pssing data as parameter

3) For deletion, print data in first Node and call deleteFromStart()

4) For displaying, call display()

void display():

1) If Size = 0, display appropriate message

2) Else

   A) Create a new Node(pointer) and set it equal to start of the List

   B) Run a loop from i = 0 to size - 1, incrementing i at each iteration

      i) Print data in pointer

      ii) Set pointer to the next Node in the List

void deleteFromStart():

1) If List is empty, display appropriate message

2) Else

   A) Set start to the next Node in the List

   B) Decrement size by 1

void insertAtEnd(int val):

1) Create a new Node with the given data

2) Increment size by 1

3) If List is empty

   A) Set Start and End to new Node

4) Else

   A) Set link of End to new Node

   B) Set end to new Node

47.2 Code

```
package linkedList;

import java.util.Scanner;

public class Queue
{

   int size;
   Node start, end;

   public void QueueRun() //For Menu-Driven usage
   {
      int choice = 0;
      int val;
      Scanner s = new Scanner(System.in);

      do
      {
         System.out.println("1 - Insertion");
         System.out.println("2 - Deletion");
         System.out.println("3 - Display Queue");
         System.out.println("9 - Exit");
         choice = s.nextInt();

         switch(choice)
         {
            case 1:    System.out.println("Enter value"); //Takes input for data
                    val = s.nextInt();
                    insertAtEnd(val); //Inserts at end of List
                    break;

            case 2:    System.out.println(start.getData()); //Prints data of first node and
                removes it
                    deleteFromStart();
                    break;

            case 3:    display(); //To Display queue
                    break;

            case 9:    System.exit(0);

            default: System.out.println("\nEnter valid Choice");
         }
      }while(choice != 9);
      s.close();
   }

   public void display()
   {
      if(size == 0)
      {
         System.out.println("Empty Queue");
         return;
      }
      else
      {
         System.out.println("\n\n\n");
         Node pointer = start;
```

```
        for(int i = 0; i < size; i++)
        {
           System.out.print(pointer.getData() + "\t");
           pointer = pointer.getLink();
        }
        System.out.println();
     }
  }

  public void deleteFromStart()
  {
     if(size == 0)
     {
        System.out.println("Queue Empty");
        return;
     }

     start = start.getLink(); //Deletes first node, sets start to next Node in list
     size--;
  }

  public void insertAtEnd(int val) //Function to insert Node at end of List
  {
     Node nptr = new Node(val, null);
     size++;

     if(start == null) //Empty List
     {
        start = nptr;
        end = start;
     }
     else
     {
        end.setLink(nptr);
        end = nptr; //Sets end to new Node
     }
  }

}
```

48 Stack (Linked List)

Write a Program to implement Stack using Linked List (Node class used directly, can be found in Other Resources)

48.1 Algorithm

1) Display Menu for user to select operation

2) For Insertion, take input of data and call insertAtStart(), pssing data as parameter

3) For deletion, print data in first Node and call deleteFromStart()

4) For displaying, call display()

void deleteFromStart():

1) If List is empty, display appropriate message

2) Else

   A) Set top to the next Node in the Stack

   B) Decrement size by 1

void display():

1) If Size = 0, display appropriate message

2) Else

   A) Create a new Node(pointer) and set it equal to top of the Stack

   B) Run a loop from i = 0 to size - 1, incrementing i at each iteration

      i) Print data in pointer

      ii) Set pointer to the next Node in the Stack

void insertAtStart(int val):

1) Create a new Node with the given data

2) Increment size by 1

3) Set link of the new Node to top

4) Set top to the new Node

48.2 Code

```
package linkedList;

import java.util.Scanner;

public class Stack
{
   int size;
   Node top;
```

```
public void StackRun() //For Menu-Driven Usage
{
   int choice = 0;
   int val;
   Scanner s = new Scanner(System.in);

   do
   {
      System.out.println("1 - Push");
      System.out.println("2 - Pop");
      System.out.println("3 - Display Stack");
      System.out.println("9 - Exit");
      choice = s.nextInt();

      switch(choice)
      {
         case 1:    System.out.println("Enter value");
                 val = s.nextInt();
                 insertAtStart(val);
                 break;

         case 2:    System.out.println(top.getData());
                 deleteFromStart();
                 break;

         case 3:    display();
                 break;

         case 9:    System.exit(0);

         default: System.out.println("\nEnter valid Choice");
      }
   }while(choice != 9);
   s.close();
}

public void insertAtStart(int val) //Function to add new Node at top of Stack
{
   Node nptr = new Node(val, null); //new Node
   size++;

      nptr.setLink(top); //Attaches Node to List
      top = nptr;
}

public void deleteFromStart() //Function to remove Node from Stack
{
   if(size == 0)
   {
      System.out.println("Stack Empty");
      return;
   }

   top = top.getLink(); //sets top to next Node
   size--;
}

public void display() //Function to display Stack
{
```

```
      if(size == 0)
      {
         System.out.println("Empty Stack");
         return;
      }
      else
      {
         System.out.println("\n\n\n");
         Node pointer = top;
         for(int i = 0; i < size; i++) //Traversal
         {
            System.out.print(pointer.getData() + "\t");
            pointer = pointer.getLink();
         }
         System.out.println();
      }
   }

}
```

49 Quick Sort (Linked List)

Write a program to sort a Linked List using QuickSort (LinkedList package used directly, included in Other Resources)

49.1 Algorithm

1) Create a LinkedList of sample data(A[ ]) or take user input (Sample data used here)

2) Call quicksort() on the LinkedList, passing the List, first and last Nodes as the parameters

3) Display the Sorted LinkedList

void quicksort(LinkedList LL, Node left, Node right):

1) Declare a pivot Node(q) and a boolean(flag) and set it to false

2) Run a loop using a Node(ptr) starting from left, till ptr is not null, setting ptr to the next Node in the list at each iteration

A) If ptr = right, raise a flag indicating right lies after left

3) If flag has been raised

A) Pass the List, left and right to partition() and assign the value to q

B) Declare a temporary Node and assign the Node preceding q to it

C) Call quicksort passing the List, left and temp as arguments (First half of list before q)

D) Call quicksort passing the List, Node after q and right as arguments (Second half of list after q)

Node partition(LinkedList LL, Node left, Node right):

1) Declare a pivot Node P and set it to left

2) Declare Nodes l and r and set them to left and right respectively

3) Declare a Node ptr and set it to the start of the Linked List for iteration

4) Run a loop from i = 0, till ptr $\neq$ left, setting ptr to the next Node in the List and incrementing i each time (to get numerical position of left)

5) Reset ptr to start of the List

6) Run a loop from j = 0, till ptr $\neq$ right, setting ptr to the next Node in the List and incrementing j each time (to get numerical position of right)

7) Create a new Node(temp)

8) Run an infinite loop

A) Run a loop till data of Node after l < data of P

i) Set l to next Node

ii) Increment i by 1

iii) If l = right, break the loop

B) Run a loop from temp = LL.start till temp $\neq$ r, setting temp to the next Node at each iteration (To set temp to node before right)

C) Run a loop till data of temp > data of P

i) Declare a Node (temp2) and set it to node before temp

ii) Set temp to temp2 (Previous Node)

iii) Decrement j by 1

iv) If temp2 = left, break the loop

D) If i ≥ j, break the loop

E) Else, swap l and temp (to sort them relative to each other)

9) If temp = left (reverse iteration reaches left), return temp

10) Swap left and temp

11) Return temp

void swap(Node a, Node b):

1) Swap the data in the two Nodes

49.2 Code

```
import linkedList.*; //To use the Node class

public class QuickSortLinkedList
{

   public static void main(String[] args)
   {
      int[] A = {11, 9, 7, 3, 63, 74, 11, 66, 78}; //Sample Data

      LinkedList LL = new LinkedList(A); //Creates a Linked List with the sample data


      QuickSortLinkedList qs = new QuickSortLinkedList(); //Class Object
      qs.quicksort(LL, LL.start, LL.end); //Call for Quicksort Method

      LL.display(); //Print Sorted List
   }

   public void quicksort(LinkedList LL, Node left, Node right) //Function to sort the List
   {
      Node q; //Pivot Node
      boolean flag = false;

      for(Node ptr = left; ptr != null; ptr = ptr.getLink()) //Loop to iterate from left
           through right to make sure right lies after left
         if(ptr == right)
            flag = true; //Indicates that the Node right has been found

      if(flag)
      {
         q = partition(LL, left, right); //Passes Nodes to partition and assigns pivot Node to q
         Node temp; //Temporary node
         for(temp = LL.start; temp != q; temp = temp.getLink()) //Loop to iterate to Node
              before q
         {}
```

```
      quicksort(LL, left, temp); //Passes first half to Sort
      quicksort(LL, q.getLink(), right); //Passes second half to sort
   }
}

public Node partition(LinkedList LL, Node left, Node right) //Function to split array into 3
    parts, left block, pivot, right block
{
   Node P = left; //Set pivot to first element in the List
   Node ptr = LL.start; //Node for iteration
   Node l = left; //First Node in the List
   Node r = right; //Last node in the part of List passed
   int i, j;  //Iteration variables

   for(i = 0; ptr != left; ptr = ptr.getLink(), i++) {} //Gets numerical position of left

   ptr = LL.start; //Resets pointer to start of List
   for(j = 0; ptr != right.getLink(); ptr = ptr.getLink(), j++) {} //Gets numerical position
       of right
   //j++;
   Node temp; //Temporary Node

   for(;;)
   {
      while(l.getLink().getData() < P.getData()) //Finds last node in the List in line which
          is smaller than Pivot
      {
         l = l.getLink(); //Gets next Node in List
         i++;  //Position
         if(l == right) //Out of bounds of part passed
            break;
      }


      for(temp = LL.start; temp != r; temp = temp.getLink()) //Sets temp to Node before right
      {}

      while(temp.getData() > P.getData()) //Finds last Node in line (from the rear) which is
          larger than Pivot
      {
         Node temp2;
         for(temp2 = LL.start; temp2 != temp; temp2 = temp2.getLink())
         {}
         temp = temp2;
         j--;

         if(temp2 == left) //Breaks if Node is at/before left
            break;
      }

      if(i >= j) //Breaks if i and j are same, or i is larger
         break;
      else
         swap(l, temp); //Swaps temp and l to sort them
   }

   if(temp == left) //if Reverse iteration reaches left, returns left-most Node
      return temp;

   swap(left, temp); //Swap left-most Node and current Node in temp
   return temp;  //Returns temp
```

```
   }

   public void swap(Node a, Node b) //Function to Swap Nodes
   {
      int temp = a.getData();
      a.setData(b.getData());
      b.setData(temp);
   }

}
```

50 Binary Tree

Write a program to implement Binary Tree. (TreeNode class and binaryTree package used directly, included in Other Resources)

Input:
1, 12, 8, 4, 6, 99, 2, 23, 74, 9, 51

Output: Number of Nodes: 11
1 12 8 4 2 6 9 99 23 74 51

50.1 Algorithm

1) Use sample data to create a Binary tree or take user input (Sample data used here)

2) Print number of Nodes

3) Perform preorder traversal of Tree

int countNodes(TreeNode root):

1) If root is null, return 0

2) Return 1 + Nodes in Left branch + Nodes in Right branch (using recursive calls)

void preorder(TreeNode root):

1) If root $\neq$ null

   A) Print data in root

   B) Call preorder, passing left branch as parameter

   C) Call preorder, passing right branch as parameter

void postorder(TreeNode root):

1) If root $\neq$ null

   A) Call postorder, passing Left branch as parameter

   B) Call postorder, passing Right branch as parameter

   C) Print data in root

void inorder(TreeNode root):

1) If root $\neq$ null

   A) Call inorder, passing Left branch as parameter

   B) Print data in root

   C) Call inorder, passing Right branch as parameter

boolean search(TreeNode root, int n):

1) If root = null, return false

2) If data in root = n, return true

3) Call search on Right and Left branches and return true if either returns true

4) Return false

void insertNode(int key):

1) Set root of BinaryTree to TreeNode returned from insertNode() when root and new TreeNode with value of key is passed as parameter

TreeNode insertNode(TreeNode currentParent, TreeNode newNode):

1) If currentParent = null, return newNode

2) If data in newNode $\geq$ data in currentParent, call insertNode, passing currentParent.right and newNode as parameters and store the value in currentParent.right

3) If data in newNode $<$ data in currentParent, call insertNode, passing currentParent.left and newNode as parameters and store the value in currentParent.left

4) Return currentParent

50.2 Code

```
package binaryTree;

public class BinaryTree
{
   TreeNode root;

   public static void main(String[] args)
   {
      int[] a = {1, 12, 8, 4, 6, 99, 2, 23, 74, 9, 51}; //Sample Data for Tree

      BinaryTree bt = new BinaryTree(a);

      System.out.println("Number of Nodes: " + bt.countNodes(bt.root)); //Prints Number of Nodes
      bt.preorder(bt.root); //Traverses the Tree (preorder)
   }

   BinaryTree() //Default Constructor
   {
      root = null;
   }

   BinaryTree(int[] a)  //Constructor to create a Tree based on given data
   {
      for(int i = 0; i < a.length; i++) //Inserts each element in Tree
         insertNode(a[i]);
   }

   int countNodes(TreeNode root) //Function to countNodes (recursive)
   {
```

```
   if(root == null) //End of branch
      return 0;
   return 1 + countNodes(root.getLeftLink()) + countNodes(root.getRightLink()); //Count
       current Node and call itself on both branches to count them
}

void preorder(TreeNode root) //Function for preorder (Root-Left-Right) traversal of Binary
    Tree (recursive)
{
   if(root != null) //Till Node exists
   {
      System.out.print(root.getData() + " "); //Prints Root
      preorder(root.getLeftLink()); //Traverses Left Tree
      preorder(root.getRightLink()); //Traverses Right Tree
   }
}

void postorder(TreeNode root) //Function for postorder (Left-Right-Root) traversal of Binary
    Tree (recursive)
{
   if(root != null) //Till Node Exists
   {
      preorder(root.getLeftLink()); //Traverses Left Tree
      preorder(root.getRightLink()); //Traverses Right Tree
      System.out.print(root.getData() + " "); //Prints Root
   }
}

void inorder(TreeNode root) //Function for inorder (Left-Root-Right) traversal of Binary
    Tree (recursive)
{
   if(root != null) //Till Node Exists
   {
      inorder(root.getLeftLink()); //Traverses Left Tree
      System.out.print(root.getData() + " "); //Prints Root
      inorder(root.getRightLink()); //Traverses Right Tree
   }
}

boolean search(TreeNode root, int n) //Function to search for a given value in Tree
{
   if(root == null) //Node does not exist
      return false;
   if(root.getData() == n) //Data exists in current Node
      return true;
   if(search(root.getLeftLink(), n) || search(root.getRightLink(), n)) //Recursive call on
       both branches of Tree
      return true;
   return false; //Default value
}


public void insertNode(int key) //Recursive function to insertNode in Tree
{
    root = insertNode(root, new TreeNode(key)); //Calls private recursive function with the
        root and a new Node
}

// private recursive call to prevent unintended additions at the wrong place

private TreeNode insertNode(TreeNode currentParent, TreeNode newNode) {
```

```
        if (currentParent == null) //Node does not exist
            return newNode; //Sets new Node here
        else if (newNode.data >= currentParent.data) //Data >= Data of Parent
            currentParent.right = insertNode(currentParent.right, newNode); //Recursive call on
                Right branch
        else if (newNode.data < currentParent.data) //Data < Data of Parent
            currentParent.left = insertNode(currentParent.left, newNode); //Recursive call on
                Left Branch

        return currentParent;
    }


}
```

51 Other Resources

51.1 Package: linkedlist

51.1.1 Class: Node

```
package linkedList;

public class Node
{
   int d;  //Data
   Node link; //Link

   public Node(int val, Node l) //Initializes the Node
   {
      d = val;
      link = l;
   }

   public void setLink(Node n)   //Modifies the Link of Node
   {
      link = n;
   }

   public void setData(int val) //Modifies the Data in Node
   {
      d = val;
   }

   public Node getLink()      //Returns the Link of the Node
   {
      return link;
   }

   public int getData()    //Returns the Data in the Node
   {
      return d;
   }

}
```

51.2 Package: binaryTree

51.2.1 Class: TreeNode

```
package binaryTree;

public class TreeNode
{
   TreeNode left, right; //Links
   int data;  //Data

   TreeNode()   //Default Contructor
   {
      left = right = null;
      data = 0;
```

```
  }

  TreeNode(int d)    //Node with data
  {
     left = null;
     right = null;
     data = d;
  }

  void setData(int d) //Function to change data
  {
     data = d;
  }

  void setLeftLink(TreeNode n) //Function to change Left Link
  {
     left = n;
  }

  void setRightLink(TreeNode n) //Function to change Right Node
  {
     right = n;
  }

  int getData() //Function to return data in a Node
  {
     return data;
  }

  TreeNode getLeftLink() //Function to return Left Link of Node
  {
     return left;
  }

  TreeNode getRightLink() //Function to return Right Link of Node
  {
     return right;
  }
}
```