Computer Science 3 –

LoanCalc –

```
/**
* Computes the periodical payment necessary to re-pay a given loan.
*/
public class LoanCalc
{

      static double epsilon = 0.001;
      static int iterationCounter;
      static int iterationCounter1;

    /**
     * Gets the loan data and computes the periodical payment.
     * Expects to get three command-line arguments: sum of the loan
(double),
     * interest rate (double, as a percentage), and number of
payments (int).
     */
      public static void main(String[] args)
      {
            // Gets the loan data
            double loan = Double.parseDouble(args[0]);
            double rate = Double.parseDouble(args[1]);
            int n = Integer.parseInt(args[2]);
            System.out.println("Loan sum = " + loan + ", interest
rate = " + rate + "%, periods = " + n);

            // Computes the periodical payment using brute force
search
            System.out.print("Periodical payment, using brute force:
");
```

```
           System.out.printf("%.2f", bruteForceSolver(loan, rate, n,
epsilon));

           System.out.println();

           System.out.println("number of iterations: " +
iterationCounter);


           // Computes the periodical payment using bisection search

           System.out.print("Periodical payment, using bi-section
search: ");

           System.out.printf("%.2f", bisectionSolver(loan, rate, n,
epsilon));

           System.out.println();

           System.out.println("number of iterations: " +
iterationCounter1);

      }


      /**

      * Uses a sequential search method  ("brute force") to compute
an approximation

      * of the periodical payment that will bring the ending balance
of a loan close to 0.

      * Given: the sum of the loan, the periodical interest rate (as
a percentage),

      * the number of periods (n), and epsilon, a tolerance level.

      */

      // Side effect: modifies the class variable iterationCounter.

    public static double bruteForceSolver(double loan, double rate,
int n, double epsilon)

      {

      // Replace the following statement with your code

           double payment = loan/n;

         double balance = loan;

         int i = 0;
```

```
        while (balance >= epsilon)
           {
            iterationCounter++;
                  payment += epsilon;
            balance = loan;
            for (i = 0; i < n; i++)
                  {
                 balance -= payment;
                 balance *= 1 + (rate/100);
            }
        }
           return payment;
    }




    /**
      * Uses bisection search to compute an approximation of the
periodical payment
      * that will bring the ending balance of a loan close to 0.
      * Given: the sum of theloan, the periodical interest rate (as
a percentage),
      * the number of periods (n), and epsilon, a tolerance level.
      */
      // Side effect: modifies the class variable iterationCounter.
    public static double bisectionSolver(double loan, double rate,
int n, double epsilon)
      {
      double L = loan/n;
           double H = loan;
           double g = ( (L + H)/2);
```

```
            while ((H - L) > epsilon )
            {
                  if ( endBalance ( loan, rate, n, g ) * endBalance (
loan, rate, n, L ) > 0 )
                  {
                        L = g;
                  }
                  else
                  {
                        H = g;
                  }
                  g = ( L + H ) / 2.0;
                  iterationCounter1++;
            }
      return g;
      }


      /**
      * Computes the ending balance of a loan, given the sum of the
loan, the periodical
      * interest rate (as a percentage), the number of periods (n),
and the periodical payment.
      */
      private static double endBalance(double loan, double rate, int
n, double payment)
      {
            // Replace the following statement with your code
            for ( int i = 0; i < n; i++ )
            {
                  loan = ( loan - payment ) * ( 1.0 + rate/100 );
            }
```

```
        return loan;
    }
}
```

LowerCase –

```
/** String processing exercise 1. */
public class LowerCase
{
    public static void main(String[] args)
     {
        String given = args[0];
           System.out.println(lowerCase ( given ));
    }
     public static String lowerCase(String word)
     {
           String result = "";
           char letter;
           int represent;
           for ( int i = 0; i < word.length(); i++ )
           {
                 letter = word.charAt(i);
                 if ( letter >= 65  &&  letter <= 90 )
                 {
                       result = result + (char)(letter + 32);
                 }
                 else
                 {
                       result = result + (char)(letter);
                 }

           }
           return result;
    }
}
```

UniqueChars –

```
 /** String processing exercise 2. */
public class UniqueChars
{
    public static void main(String[] args)
      {
        String str = args[0];
        System.out.println(uniqueChars(str));
    }
    public static String uniqueChars(String word)
      {
        String newWord = "";
           for ( int i = 0; i < word.length(); i++ )
           {
                boolean check = false;
                char letter = word.charAt(i);
                for ( int j = 0; j <newWord.length(); j++ )
                {
                     char letter2 = newWord.charAt(j);
                     if ( letter == letter2 )
                     {
                          check = true;
                     }
                }
                if ( check == false || letter == 32  )
                {
                     newWord = newWord + letter;
                }
           }
           return newWord;
        }
    }
```

Calendar0 –

```
/*
 * Checks if a given year is a leap year or a common year,
 * and computes the number of days in a given month and a given
year.
 */
public class Calendar0
{

	// Gets a year (command-line argument), and tests the
functions isLeapYear and nDaysInMonth.
	public static void main(String args[])
	{
		int year = Integer.parseInt(args[0]);
		isLeapYearTest(year);
		nDaysInMonthTest(year);
	}

	// Tests the isLeapYear function.
	private static void isLeapYearTest(int year)
	{
		String commonOrLeap = "common";
		if (isLeapYear(year))
		{
			commonOrLeap = "leap";
		}
		System.out.println(year + " is a " + commonOrLeap + "
year");
	}

	// Tests the nDaysInMonth function.
	private static void nDaysInMonthTest(int year)
	{
		if ( (year % 4 == 0 && year % 100 != 0 ) || ( year % 400
== 0 ))
		{
			System.out.println( "Month 1 has 31 days" );
			System.out.println( "Month 2 has 29 days" );
			System.out.println( "Month 3 has 31 days" );
			System.out.println( "Month 4 has 30 days" );
			System.out.println( "Month 5 has 31 days" );
			System.out.println( "Month 6 has 30 days" );
			System.out.println( "Month 7 has 31 days" );
			System.out.println( "Month 8 has 31 days" );
			System.out.println( "Month 9 has 30 days" );
			System.out.println( "Month 10 has 31 days" );
			System.out.println( "Month 11 has 30 days" );
			System.out.println( "Month 12 has 31 days" );
		}
		else
		{
			System.out.println( "Month 1 has 31 days" );
			System.out.println( "Month 2 has 28 days" );
			System.out.println( "Month 3 has 31 days" );
			System.out.println( "Month 4 has 30 days" );
			System.out.println( "Month 5 has 31 days" );
			System.out.println( "Month 6 has 30 days" );
			System.out.println( "Month 7 has 31 days" );
			System.out.println( "Month 8 has 31 days" );
			System.out.println( "Month 9 has 30 days" );
			System.out.println( "Month 10 has 31 days" );
```

```
                System.out.println( "Month 11 has 30 days" );
                System.out.println( "Month 12 has 31 days" );
            }

        }

        // Returns true if the given year is a leap year, false
otherwise.
        public static boolean isLeapYear(int year)
        {
           boolean isleap;
            if ( (year % 4 == 0 && year % 100 != 0 ) || ( year % 400
== 0 ))
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        // Returns the number of days in the given month and year.
        // April, June, September, and November have 30 days each.
        // February has 28 days in a common year, and 29 days in a
leap year.
        // All the other months have 31 days.
        public static int nDaysInMonth(int month, int year)
        {
            if ( (year % 4 == 0 && year % 100 != 0 ) || ( year % 400
== 0 ))
```

```
           {
                if ( month == 1 || month == 3 || month == 5 || month
== 7 || month == 8 || month == 10 || month == 12 )
                {
                      month = 31;
                }
                else if ( month == 2 )
                {
                      month = 29;
                }
                else
                {
                      month = 30;
                }
           }
           else
           {
                if ( month == 1 || month == 3 || month == 5 || month
== 7 || month == 8 || month == 10 || month == 12 )
                {
                      month = 31;
                }
                else if ( month == 2 )
                {
                      month = 28;
                }
                else
                {
                      month = 30;
                }
           }
```

```
        return month;
    }
}
```

Calendar –

```
public class Calendar
{

      static int dayOfMonth = 1;
      static int month = 1;
      static int year = 1900;
      static int dayOfWeek = 2;
      static int nDaysInMonth = 31;
      public static void main(String args[])
      {
            String str = "";
            int pickedY = Integer.parseInt(args[0]);
            while (year < pickedY + 1)
            {
                  advance();
                  while(year == pickedY)
                  {
                        str = "";
                        if(dayOfWeek == 1) str = " Sunday";
                        System.out.println(dayOfMonth + "/" + month +
"/" + year + str);
                        advance();
                  }
            }
      }
      private static void advance()
      {
            if(dayOfWeek == 7) dayOfWeek = 1;
            else dayOfWeek++;
```

```
		if(dayOfMonth == nDaysInMonth)
		{
			if(month == 12)
			{
				month = 0;
				year++;
			}
			dayOfMonth = 1;
			month++;
			nDaysInMonth = nDaysInMonth(month, year);
		}
		else dayOfMonth++;
	}
	private static boolean isLeapYear(int year)
	{
		if(year % 4 == 0 && (year % 100 != 0 || year % 400 == 0))
		{
			return true;
		}
		else
		{
			return false;
		}
	}
	private static int nDaysInMonth(int month, int year)
	{
		if (((month == 4) || (month == 6) || (month == 9) ||
(month == 11)))
		{
			return 30;
```

```
            }
            else if ((month == 1) || (month == 3) || (month == 5) ||
(month == 7) || (month == 8) || (month == 10) || (month == 12))
return 31;
            else if(isLeapYear(year)) return 29;
            else return 28;
      }
}
```