```
public class LoanCalc {
        static double epsilon = 0.001;  // The computation tolerance (estimation error)
        static int iterationCounter;    // Monitors the efficiency of the calculation


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


                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);


                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: " + iterationCounter);
        }
   public static double bruteForceSolver(double loan, double rate, int n, double
epsilon) {
        // Replace the following statement with your code
                double payment = (loan / n);
                while(endBalance(loan, rate, n, payment) > 0){
                        payment = payment + epsilon;
                        iterationCounter++;
                }
                return payment;
   }
   public static double bisectionSolver(double loan, double rate, int n, double epsilon)
{
        iterationCounter = 0;
                double high = loan;
                double low = (loan / n);
                double payment = ((high + low) / 2);
                while((high - low) > epsilon) {
                        if((endBalance(loan, rate, n, payment)) * (endBalance(loan, rate,
n ,low)) > 0)
                                low = payment;
                        else
                                high = payment;
                        payment = ((high + low) / 2);
                        iterationCounter++;
                }
                return payment;


   }
        private static double endBalance(double loan, double rate, int n, double
payment) {
                // Replace the following statement with your code
                double endBal = loan;
                        for(int i = 0; i < n; i++) {
                                endBal = ((endBal - payment) * (1 + (rate / 100)));
                        }
                        return endBal;
                }


}
```

```
public class LowerCase {
    public static void main(String[] args) {
        String str = args[0];
        System.out.println(lowerCase(str));
    }

    public static String lowerCase(String str) {
        // Replace the following statement with your code
                    String newStr = "";
                    char newChar;
                    int asci;
                    for(int i = 0; i < str.length(); i++){
                              newChar = str.charAt(i);
                              if((newChar >= 'A') && (newChar <= 'Z')){
                                        asci = (int) (newChar);
                                        asci = asci + 32;
                                        newChar = (char) (asci);
                              }
                              newStr = newStr + newChar;
                    }

        return newStr;
    }
}
```

```
public class UniqueChars {
   public static void main(String[] args) {
      String str = args[0];
      System.out.println(uniqueChars(str));
   }

   public static String uniqueChars(String s) {
      String newStr = "" + s.charAt(0);
                boolean exist = false;
                for(int i = 1; i < s.length(); i++){
                        if(s.charAt(i) == ' ') newStr = newStr + s.charAt(i);
                        else{
                        for(int j = 0; j < newStr.length(); j++){
                                if(newStr.charAt(j) == s.charAt(i)) exist = true;
                        }
                        if (!exist){
                                newStr = newStr + s.charAt(i);
                        }

                }
                exist = false;

   }
        return newStr;
}
}
```

```
    public class Calendar {
        // Starting the calendar on 1/1/1900
        static int dayOfMonth = 1;
        static int month = 1;
        static int year = 1900;
        static int dayOfWeek = 2;     // 1.1.1900 was a Monday
        static int nDaysInMonth = 31; // Number of days in January

        /**
         * Prints the calendars of all the years in the 20th century. Also prints
the
         * number of Sundays that occured on the first day of the month during
this period.
         */
        public static void main(String args[]) {
        int whichYear = Integer.parseInt(args[0]);

            // Advances the date and the day-of-the-week from 1/1/1900 till
31/12/1999, inclusive.
            // Prints each date dd/mm/yyyy in a separate line. If the day is a
Sunday, prints "Sunday".
            // The following variable, used for debugging purposes, counts
how many days were advanced so far.
            int debugDaysCounter = 0;
            //// Write the necessary initialization code, and replace the
condition
            //// of the while loop with the necessary condition
            while (year < whichYear)
                advance();
            while (year == whichYear) {
                //// Write the body of the while
                if(dayOfWeek == 1){
                    System.out.println(dayOfMonth + "/" + month + "/" + year + " Sunday");
                }
                else{
                    System.out.println(dayOfMonth + "/" + month + "/" + year);
                }



                advance();
                debugDaysCounter++;
                //// If you want to stop the loop after n days, replace the condition of the
                //// if statement with the condition (debugDaysCounter == n)
                if (false) {
                    break;
                }

            }
            //// Write the necessary ending code here
        }

        // Advances the date (day, month, year) and the day-of-the-week.
        // If the month changes, sets the number of days in this month.
        // Side effects: changes the static variables dayOfMonth, month, year, dayOfWeek, nDaysInMonth.
        private static void advance() {
            // Replace this comment with your code
            if(dayOfWeek == 7) dayOfWeek = 1;
            else dayOfWeek++;
            if(dayOfMonth == nDaysInMonth(month, year)){
```

```
        if(month == 12) {
            month = 1;
            year++;
            dayOfMonth = 1;
        }
        else{
            month++;
            dayOfMonth = 1;


        }
    }
    else{
        dayOfMonth++;
 }
 }
// Returns true if the given year is a leap year, false otherwise.
private static boolean isLeapYear(int year) {
    // Replace the following statement with your code
    boolean isLeap;
    isLeap = ((year % 400) == 0);
    isLeap = isLeap || (((year % 4) == 0) && ((year % 100) != 0));
    return isLeap;
}

// 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.
private static int nDaysInMonth(int month, int year) {
    // Replace the following statement with your code
```

```
            if(month == 4 || month == 6 || month == 9 || month ==11) return 30;
            if(month == 1 || month == 3 || month == 4 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12) return 31;
            if(month == 2){
                if(isLeapYear(year)) return 29;
                else return 28;
            }
            else return 0;
        }
    }
```