LowerCase

```
public class LowerCase
{
   public static void main(String[] args)
   {
      String str = args[0];
      System.out.println(lowerCase(str)); //printing the new string after low his
letter
   }

   public static String lowerCase(String s)
   {
      String newone="";
      for(int i = 0;i<s.length(); i++)
      {
         char newchar ='a';
         int valueofchar=s.charAt(i);//getting the value of the char
         if(64<valueofchar&&valueofchar<91)//check if the letter is Capital
          newchar = (char)(valueofchar+32 );
            else  newchar = (char)(valueofchar);//32 is the diffrense between
capitl and low letter
         newone+= newchar;



      }


      return newone;
   }
}
```

UniqueChars

```
public class UniqueChars
 {
    public static void main(String[] args)
     {
        String str = args[0];
        System.out.println(uniqueChars(str));//printing the new string after
changing
    }
    public static String uniqueChars(String s)
    {
        String newstring= "" + s.charAt(0);
        for(int i = 0;i<s.length(); i++)
        {
            char checkdup=s.charAt(i);
            if(checkdup==' ') //checking if there is space and add
               {
                  newstring+= " ";
               }
                else //if the char is not space
               {
                  boolean ifdup=true;
                  for(int j = 0;j<newstring.length(); j++)
                  {
                       if(checkdup==newstring.charAt(j))//if the char exsist than i will
not add the char to the string
                        ifdup= false;
                   }
                   if (ifdup)//if the string not exsist add it
                     newstring+=checkdup;
               }
```

```
        }
        return newstring;
    }

}
```

LoanCalc

```
public class LoanCalc
{

	static double epsilon = 0.001;  // The computation tolerance (estimation
error)
	static int iterationCounter;
	static int iterationCounter1;   // Monitors the efficiency of the calculation


	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);
	}
```

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


                double payment = loan/n;//deter the first payment to be
loan/period
                while (endBalance(loan,rate,n,payment) >= 0)
                {
                        payment += epsilon;//advence the payment with epsioln
                        iterationCounter++; //check how much time the loop run
                }
                 return payment; // returning the payment
   }



   public static double bisectionSolver(double loan, double rate, int n, double
epsilon)
   {



                double payment =  loan/n, loanforwork = loan;//stetting the first
payment to be loan/period
                double checkpayment = (payment + loan) /2; //setting g
according to algoritem
                while (loanforwork - payment > epsilon)
                 {
                        if (endBalance(loan,rate,n,checkpayment) *
endBalance(loan,rate,n,payment)>0 ) //if f(g)*f(l)>0
                                payment = checkpayment;
                                else
                                        loanforwork = checkpayment;
                        checkpayment = (loanforwork + payment) / 2;
```

```
                    iterationCounter1++; //checking how much time the loop
run
                }
                return checkpayment; //returning the payment
    }


        private static double endBalance(double loan, double rate, int n, double
payment)
        {
                double endofbalance = loan;
                for (int i=0;i<n;i++)
                        endofbalance = (endofbalance-payment) * (1+rate/100);
//cheking how much mony left to pay
                return endofbalance; // returning the rest of the mony that left
after the payments


        }
}
```

Calendar0

```
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) {
		for(int i = 1;i<13;i++)
			System.out.println("Month "+i+ " has "+
nDaysInMonth(i,year) +" days");
	}

	// Returns true if the given year is a leap year, false otherwise.
	public static boolean isLeapYear(int year) {
```

```
        boolean isleap = false;
    if(year%4==0)//check if the year davise by 4
    {
        isleap=true;
        if (year%400!=0&&year%100==0)//check if the year deviseble by
400 but not 100
            isleap = false;



    }
        return isleap;
}


public static int nDaysInMonth(int month, int year)
{
        if (month == 4 ||month == 6 ||month == 9 ||month == 11)// checkif
the month is april june september or november
        return 30;
        if(month==2)//check if the month is feb and check how many
days it should have depend on either it is leap ot common year
        {
            if (isLeapYear(year))
                    return 29;
         return 28;
        }
        return 31;//if the month is nither of the month above


}
}
```

Calendar1

```
public class Calendar1
{
   // 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

        public static void main(String args[])
        {


                        advance();



         }

         private static void advance()
         {
                int countdate = 0;
                int day = 2;
                for(int year = 1900;year<2000;year++)
                {
                        for (int i =1;i<13;i++)
                        {
                                for (int j=1;j<nDaysInMonth(i,year)+1;j++)
                                {
                                        if(day>7)//check if the count is over 7 and
than start counting again
```

```
                                        day=1;
                                    if(day==1)//check if sunday
                                    {
                                        if(j==1)//check if sunday is the first
day of the month
                                            countdate++;
                                        System.out.println(j+"/"+i+"/"+year+"
Sunday");
                                    }
                                    else  System.out.println(j+"/"+i+"/"+year);
                                    day++;
                                }


                            }
                        }
                        System.out.println("During the 20th century, " +countdate+"
Sundays fell on the first day of the month");




                }


    // Returns true if the given year is a leap year, false otherwise.
            private static boolean isLeapYear(int year) {
                boolean isleap = false;
                if(year%4==0)//check if the year davise by 4
                {
                    isleap=true;
                    if (year%400!=0&&year%100==0)//check if the year deviseble by
400 but not 100
                            isleap = false;
```

```
            }
                return isleap;
        }

        private static int nDaysInMonth(int month, int year) {
                if (month == 4 ||month == 6 ||month == 9 ||month == 11)// checkif the month is april june september or november
                return 30;
                if(month==2)//check if the month is feb and check how many days it should have depend on either it is leap ot common year
                {
                        if (isLeapYear(year))
                                        return 29;
                 return 28;
                }
                return 31;//if the month is nither of the month above

        }
}
```

Calendar

```
public class Calendar
{
   // Starting the calendar on 1/1/1900
        static int dayOfMonth = 1;
        static int month = 1;
        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 year = Integer.parseInt(args[0]);
                        advance(year);//initiate the function with the givven year



      }


         private static void advance(int years)
         {
                int day = 2;
                for(int year = 1900;year<years;year++)
                {
                        for (int i =1;i<13;i++)
                        {
```

```
                        for (int
j=1;j<nDaysInMonth(i,year)+1;j++)//checking when is sunday
                        {
                                if(day>7)
                                        day=1;
                                day++;
                        }

                }
        }
                for (int i =1;i<13;i++)
                {
                        for (int j=1;j<nDaysInMonth(i,years)+1;j++)
                        {
                                if(day>7)//check if the week is passed
                                        day=1;
                                if(day==1)//check if sunday
                                {
                                        System.out.println(j+"/"+i+"/"+years+"
Sunday");
                                }
                                else  System.out.println(j+"/"+i+"/"+years);
                                day++;
                        }

                }


    }

  // Returns true if the given year is a leap year, false otherwise.
```

```
    private static boolean isLeapYear(int year) {
        boolean isleap = false;
        if(year%4==0)//check if the year davise by 4
        {
            isleap=true;
            if (year%400!=0&&year%100==0)//check if the year deviseble by
400 but not 100
                isleap = false;



        }
            return isleap;
    }


    private static int nDaysInMonth(int month, int year) {
            if (month == 4 ||month == 6 ||month == 9 ||month == 11)// checkif
the month is april june september or november
            return 30;
            if(month==2)//check if the month is feb and check how many
days it should have depend on either it is leap ot common year
            {
                if (isLeapYear(year))
                        return 29;
             return 28;
            }
            return 31;//if the month is nither of the month above
    }
}
```