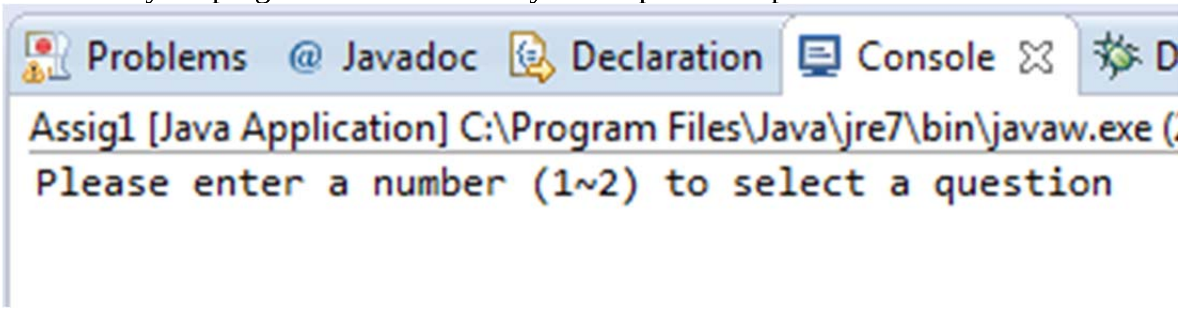
CSCI2010 Principle of Computer Science

Laboratory Seven

Activity 0: Preparation

1. Create a project called “lab7” under Eclipse;
2. Copy “lab7” into project;
3. Copy “questions” into project;

Make sure your program can run correctly. An expected output of the UI is shown below.



```
Assig1 [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (
Please enter a number (1~2) to select a question
```

Activity 1: Counting and Summing Digits(2 marks)

The problem of counting or summing the digits contained in a positive integer can be solved recursively. For example, to count the number of digits, think as follows:

1. If the integer is less than 10 there is only one digit (the base case).
2. Otherwise, the number of digits is 1 (for the units digit) plus the number of digits in the rest of the integer (what's left after the units digit is taken off). For example, the number of digits in 3278 is 1 + the number of digits in 327.
3. The following is the recursive algorithm implemented in Java. (You can find the following code in *questions.java*)

===========================Code Example==============================

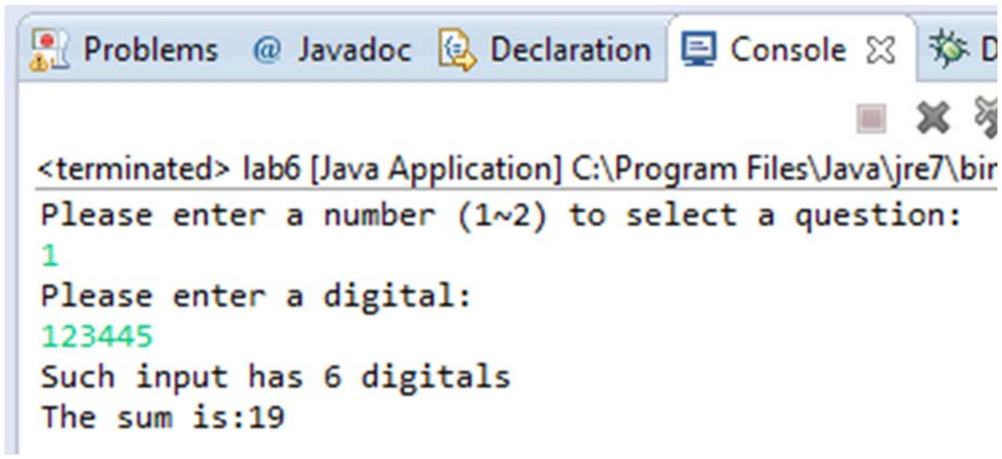
```
public int numDigits (int num)
{
  if (num < 10)
    return (1);   // a number < 10  has only one digit
  else
    return (1 + numDigits (num / 10));
}
```

=========================End of Code Example===========================

Note that in the recursive step, the value returned is 1 (counts the units digit) + the result of the call to determine the number of digits in num / 10. Recall that num/10 is the quotient when num is divided by 10 so it would be all the digits except the units digit.

In *questions.java*, you will find a method named *sumDigits*, finish this method so that it can find the sum of the digits in a positive integer. The algorithm for *sumDigits* is very similar to *numDigits*; you only have to change two lines!

An expected output is shown below:



```
<terminated> lab6 [Java Application] C:\Program Files\Java\jre7\bir
Please enter a number (1~2) to select a question:
1
Please enter a digital:
123445
Such input has 6 digitals
The sum is:19
```

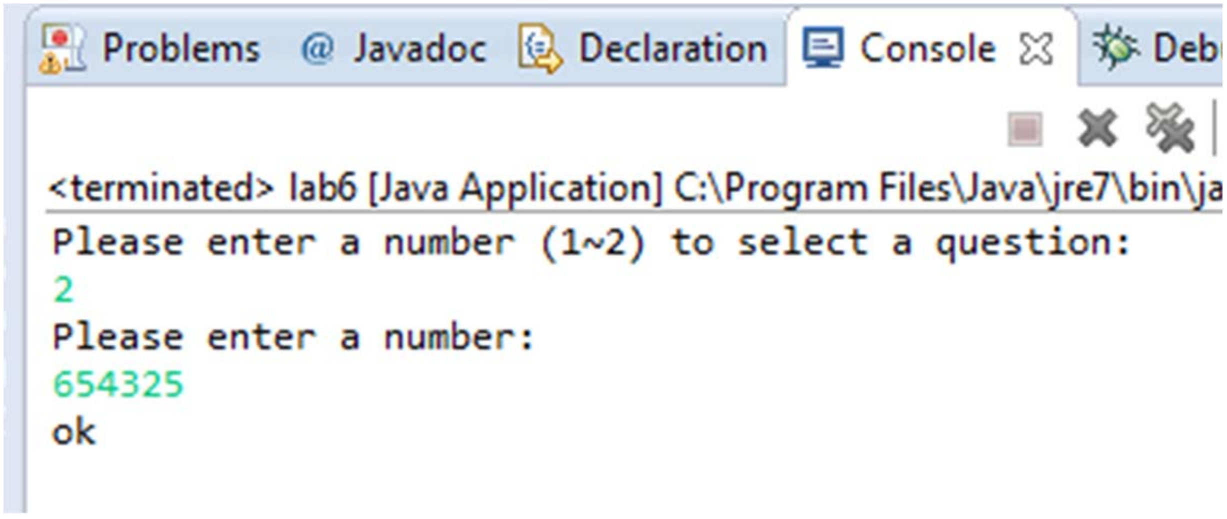
Activity 2: Counting and Summing Digits (1 marks)

Most identification numbers, such as the ISBN number on books or the Universal Product Code (UPC) on grocery products or the identification number on a traveler's check, have at least one digit in the number that is a check digit. The check digit is used to detect errors in the number. The simplest check digit scheme is to add one digit to the identification number so that the sum of all the digits, including the check digit, is evenly divisible by some particular integer. For example, American Express Traveler's checks add a check digit so that the sum of the digits in the id number is evenly divisible by 9. United Parcel Service adds a check digit to its tracking numbers such that a weighted sum of the digits (some of the digits in the number are multiplied by numbers other than 1) is divisible by 7.

For example:

654325---ok	1800237 --- ok
3429072 --- error	3180012 --- error

In *q2*, please test your *sumDigits* method to do the following: input an identification number (a positive integer), then determine if the sum of the digits in the identification number is correct. An expected output is shown below:



```
<terminated> lab6 [Java Application] C:\Program Files\Java\jre7\bin\ja
Please enter a number (1~2) to select a question:
2
Please enter a number:
654325
ok
```

Activity 3: Base Conversion (3 marks)

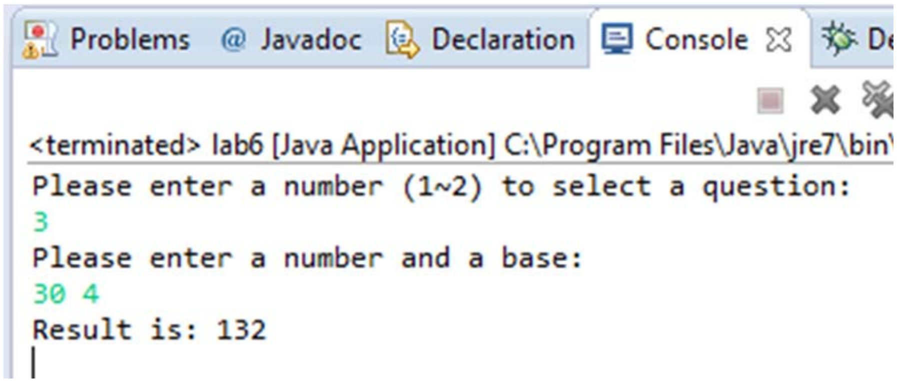
In *questions.java*, a method called “*convert*” is defined, please finish the method body that uses recursion to convert a base 10 number into a base b number, where b<10. If the number to be converted is n, then the algorithm to convert n to base b is:

1. Divide n by b. Store the quotient and the reminder.
2. The reminder is the rightmost digit of the final answer.
3. The quotient is now the new number n that you will recursively convert to base b.
4. Repeat step 1 by calling your recursive method with quotient and the original base b.
5. Stop when n/b=0. The remainder at this point will be the first digit of the final answer.

For example, to convert 30 into base 4 number:

number	Quotient	Reminder
30/4	7	2
7/4	1	3
1/4	0	1

The answer is the remainder column read bottom to top, so 30 (base 10) => 132(base 4).



```
<terminated> lab6 [Java Application] C:\Program Files\Java\jre7\bin\
Please enter a number (1~2) to select a question:
3
Please enter a number and a base:
30 4
Result is: 132
```

Activity 4: The Ackermann Function (4 marks)

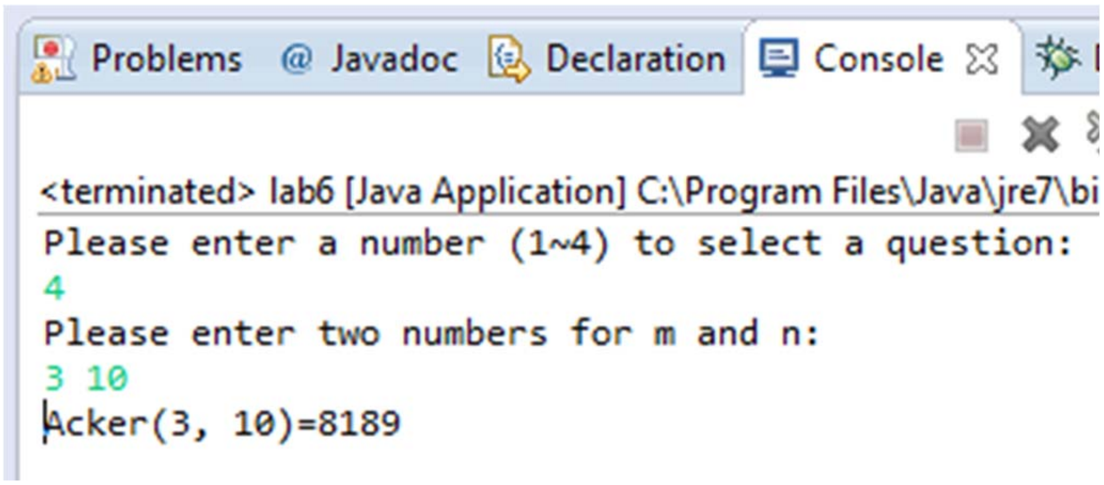
The Ackermann function is a function created by Wilhelm Ackermann in the late 1920s. For non-negative integers m and n, the Ackermann function is defined as

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1,\ A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0 \end{cases}$$

In *questions.java*, a method called "*Acker*" is defined, now finish the method body so that it can recursively computer the Ackermann function. Note that the Ackermann function grows extremely quickly for even small values of m and n. Test your method with the following values but be aware that if m is greater than 3 and n is greater than 1, it will take a very long time to compute.

	n										
m	0	1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11	12
2	3	5	7	9	11	13	15	17	19	21	23
3	5	13	29	61	125	253	509	1021	2045	4093	8189

An expected output is shown below:



```
<terminated> lab6 [Java Application] C:\Program Files\Java\jre7\bi
Please enter a number (1~4) to select a question:
4
Please enter two numbers for m and n:
3 10
Acker(3, 10)=8189
```

What should be submitted?

Please submit the following code files

- questions.java

============================The End of Lab7============================