**[CSE 1310](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/index.html) -** [**Assignments**](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/assignments/index.html) **- Programming Assignment 4**

The assignment will be graded out of 100 points.

Some tasks ask you to write code, and specify what name to use for the file where you save that code. You need to use exactly the name that is given (do not change the case, or make any other modification). Remember, the name of the main class must match the filename.

For some tasks you need to answer questions. Create a text document entitled answers.txt, or answers.docx, or answers.pdf, and put all your answers there. Acceptable file formats are plain text, Word document, OpenOffice document, and PDF. Put your name and UTA ID in the file on the first line.

Each task below will instruct you where to put your answers.

**Task 1 (10 pts.)**

public class task1

{

public static void main(String[] args)

{

int a = 24;

int b = 42;

while (b != 0)

{

int t = b;

b = a % b;

a = t;

System.out.printf("a = %3d, b = %3d\n", a, b);

}

}

}

If you execute this program, what will be printed? Put your answer in your answers file.

**Task 2 (10 pts.)**

public class task2

{

public static void main(String[] args)

{

int a = 3;

int b = 6;

while (a < b)

{

a += 2;

b++;

System.out.printf("a = %d, b = %d\n", a, b);

}

}

}

If you execute this program, what will be printed? Put your answer in your answers file.

**Task 3 (10 pts.)**

public class task3

{

public static void main(String[] args)

{

int a = 65;

int limit = 50;

while (a != limit)

{

a = limit - (limit-a)/2;

System.out.printf("a = %2d\n", a);

}

}

}

If you execute this program, what will be printed? Put your answer in your answers file.

**Task 4 (10 pts.)**

The extraterrestrials living in the planet Numerion revere a specific type of integers N. In particular, for those creatures, an integer N is holy if there exists an integer i >= 1 such that:

N = i\*i + i

Here are some examples of holy numbers:

2 (because 2 = 1\*1 + 1)

6 (because 6 = 2\*2 + 2)

12 (because 12 = 3\*3 + 3)

In a file called task4.java, write a program that:

* Asks the user to enter an integer N.
* Prints out whether that number is a holy number in planet Numerion.

For example: if the user enters 25, your program output should look EXACTLY like this:

Enter an integer N: 25

25 is not a holy number in Numerion.

Exiting...

As another example: if the user enters 12 (a holy number, since 12 = 3\*3 + 3), your program output should look EXACTLY like this:

Enter an integer N: 12

12 is a holy number in Numerion.

Exiting...

Your program's output should match EXACTLY the format shown above. There should be no deviations, no extra spaces or lines, no extra punctuation in your output. What you see above as uppercase letters should remain uppercase in your output, what you see as lowercase letters should remain as lowercase in your output, what you see as spaces and punctuation should remain exactly as spaces and punctuation in your output.

**Hint:** This is a smoking gun problem, like determining if a number is prime. You may find it useful to use ideas from the [prime\_numbers\_while.java](http://vlm1.uta.edu/%7Eathitsos/courses/cse1310_fall2015/codebase/06a_loops/prime_numbers_while.java) program we wrote in class, that used a while loop to determine if a number is prime.

**Task 5 (10 pts.)**

In a file called task5.java, write a program that:

* Asks the user to enter a word of text.
* Prints that word, with all vowels removed.

For example: if the user enters "Arlington", your program output should look EXACTLY like this:

Enter a word: Arlington

rlngtn

Exiting...

As another example: if the user enters "mountain", your program output should look EXACTLY like this:

Enter a word: mountain

mntn

Exiting...

Your program's output should match EXACTLY the format shown above. There should be no deviations, no extra spaces or lines, no extra punctuation in your output. What you see above as uppercase letters should remain uppercase in your output, what you see as lowercase letters should remain as lowercase in your output, what you see as spaces and punctuation should remain exactly as spaces and punctuation in your output.

**Task 6 (10 pts.)**

In a file called task6.java, write a program satisfying these specs:

* The program asks the user to enter an integer called "date", indicating the date of the first Sunday in March.
* If the user enters a date such that date < 1, or date > 7, then the program prints "invalid entry", and terminates.
* Otherwise (if 1 ≤ date ≤ 7), the program prints the dates of all Sundays in March, based on what the user enters.

Remember that March has 31 days.

For example, if the user enters 5, your program output should look EXACTLY like this:

When is the first Sunday this March? 5

This March, Sundays fall on:

March 5

March 12

March 19

March 26

As another example, if the user enters 10, your program output should look EXACTLY like this:

When is the first Sunday this March? 10

invalid entry

Your program's output should match EXACTLY the format shown above. There should be no deviations, no extra spaces or lines, no extra punctuation in your output. What you see above as uppercase letters should remain uppercase in your output, what you see as lowercase letters should remain as lowercase in your output, what you see as spaces and punctuation should remain exactly as spaces and punctuation in your output.

**Task 7 (10 pts.)**

In a file called task7.java, write a program write a program satisfying these specs:

* It asks the user to enter an integer called "low".
* It asks the user to enter an integer called "high".
* If low ≤ high, the program prints, in ascending order, all integers between low and high (including low and high), **except for** multiples of 4.
* If low > high, the program just prints "no numbers found" and exits.

For example, if the user enters 7 for low and 13 for high, your program output should look EXACTLY like this:

Enter low: 7

Enter high: 13

7

9

10

11

13

As another example, if the user enters 5 for low and 4 for high, your program output should look EXACTLY like this:

Enter low: 5

Enter high: 4

no numbers found

Your program's output should match EXACTLY the format shown above. There should be no deviations, no extra spaces or lines, no extra punctuation in your output. What you see above as uppercase letters should remain uppercase in your output, what you see as lowercase letters should remain as lowercase in your output, what you see as spaces and punctuation should remain exactly as spaces and punctuation in your output.

**Task 8 (10 pts.)**

In a file called task8.java, write a program that:

* Asks the user to enter a positive integer N > 1.
* If N <= 1, the program terminates.
* If N > 1, the program prints out N0, N1, N2, ..., up to (and including) the last power of N that is less than or equal to 40000.

For example: if the user enters 200, your program output should look EXACTLY like this:

Enter a positive integer N > 1: 200

1

200

40000

Exiting...

As another example: if the user enters 0, your program output should look EXACTLY like this:

Enter a positive integer N > 1: 0

Exiting...

As another example: if the user enters 3, your program output should look EXACTLY like this:

Enter a positive integer N > 1: 3

1

3

9

27

81

243

729

2187

6561

19683

Exiting...

Your program's output should match EXACTLY the format shown above. There should be no deviations, no extra spaces or lines, no extra punctuation in your output. What you see above as uppercase letters should remain uppercase in your output, what you see as lowercase letters should remain as lowercase in your output, what you see as spaces and punctuation should remain exactly as spaces and punctuation in your output.

**Task 9 (10 pts.)**

In a file called task9.java, write a program that:

* Asks the user to enter a positive integer N > 0.
* If N <= 0, the program terminates.
* If N > 0, the program prints out:
  + A line with one character '\*'.
  + A line with two characters '\*'.
  + A line with three characters '\*'.
  + And so on, up to and including a line with N characters '\*'.

For example: if the user enters 5, your program output should look EXACTLY like this:

Enter a positive integer N > 0: 5

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

Exiting...

As another example: if the user enters 0, your program output should look EXACTLY like this:

Enter a positive integer N > 0: 0

Exiting...

As another example: if the user enters 1, your program output should look EXACTLY like this:

Enter a positive integer N > 0: 1

\*

Exiting...

Your program's output should match EXACTLY the format shown above. There should be no deviations, no extra spaces or lines, no extra punctuation in your output. What you see above as uppercase letters should remain uppercase in your output, what you see as lowercase letters should remain as lowercase in your output, what you see as spaces and punctuation should remain exactly as spaces and punctuation in your output.

**Task 10 (10 pts.)**

In a file called task10.java, write a program that:

* Asks the user to enter an integer M.
* Asks the user to enter an integer N.
* Prints out the number of multiples of 11 that are greater than or equal to M and less than or equal to N.

For example: if the user enters 10 and 40, the multiples of 11 between those two numbers are 11, 22, 33, and your program output should look EXACTLY like this:

Enter an integer M: 10

Enter an integer N: 40

3 numbers between 10 and 40 are multiples of 11.

Exiting...

As another example: if the user enters 40 and 10, there are no multiples of 11 that are >= 40 and <= 10, and your program output should look EXACTLY like this:

Enter an integer M: 40

Enter an integer N: 10

0 numbers between 40 and 10 are multiples of 11.

Exiting...

As another example: if the user enters 70 and 110, the multiples of 11 between those numbers are 77, 88, 99, 110, and your program output should look EXACTLY like this:

Enter an integer M: 70

Enter an integer N: 110

4 numbers between 70 and 110 are multiples of 11.

Exiting...

Your program's output should match EXACTLY the format shown above. There should be no deviations, no extra spaces or lines, no extra punctuation in your output. What you see above as uppercase letters should remain uppercase in your output, what you see as lowercase letters should remain as lowercase in your output, what you see as spaces and punctuation should remain exactly as spaces and punctuation in your output.

**Suggestions**

Pay close attention to all specifications on this page, including file names and submission format. Even in cases where the program works correctly, points will be taken off for non-compliance with the instructions given on this page (such as wrong file names, wrong compression format for the submitted code, and so on). The reason is that non-compliance with the instructions makes the grading process significantly (and unnecessarily) more time consuming. Contact the instructor or TA if you have any questions.

**How to submit**

The assignment should be submitted via [Blackboard](http://elearn.uta.edu). Submit a ZIPPED directory called assignment4.zip (no other forms of compression accepted, contact the instructor or TA if you do not know how to produce .zip files).

To create a zipped directory called assignment4.zip, follow these steps:

1. Create a folder called assignment4.
2. Copy to that folder all your solutions (your answers file, and all your Java files).
3. Zip that folder. On windows, you can zip a folder by right-clicking on the folder, and then selecting Send to->Compressed (zipped) folder.

Your zip file should contain only 8 files: your answers document and all the Java code files (task4.java, task5.java, task6.java, task7.java, task8.java, task9.java, task10.java).

**Submission checklist**

* Did you create the answers file with your name, UTA ID, and answers to non-programming tasks?
* Did you zip everything into a file called assignment4.zip?