Programming Assignment 2

Due Thu. Jun 18, 11:55 pm

Problem 1 – Lebanese Flag

Below are two ASCII Art pictures of the Lebanese flag. ASCII art is a graphic design technique that consists of pictures pieced together from the 95 printable characters defined by the ASCII Standard. The flag to the left is composed of 32 rows and 65 columns. The flag to the right is composed of 24 rows and 49 columns. In general, the number of columns will always be twice the number of rows plus 1.

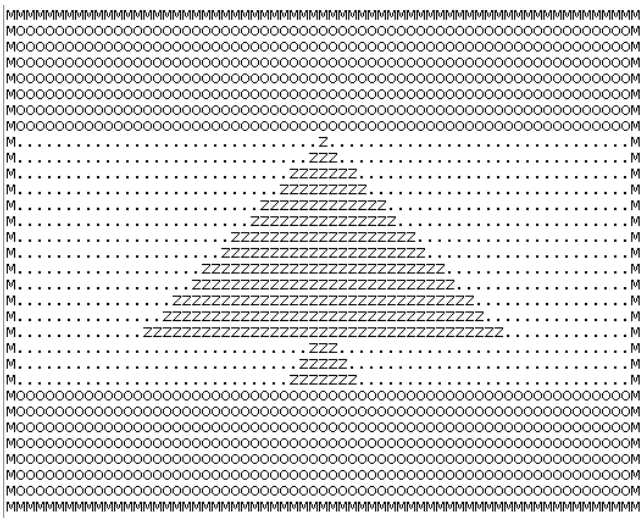


Figure 1: 32x65

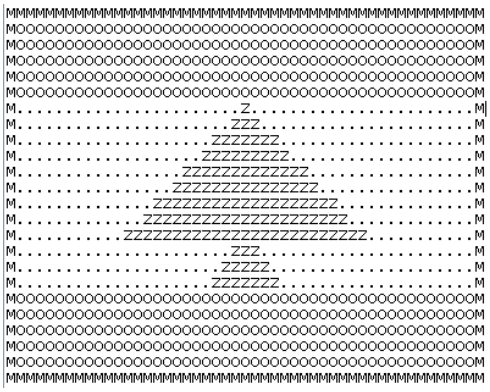


Figure 2: 24x49

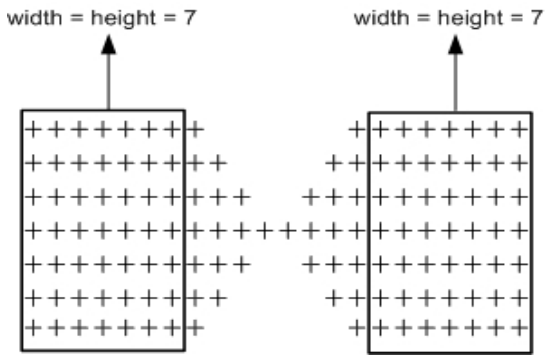
The ASCII Art pictures of the Lebanese flag have the following restrictions and details:

1. The number of rows, r, should be a multiple of 4.
2. 25% of r should be allocated for the upper part of the flag, 25% of r should be allocated for the lower part of the flag, and the remaining 50% should be allocated for the middle part of the flag.
3. The boundaries of the flag should be composed of the letter 'M'.
4. The tree part of the flag should be composed of the letter 'Z'. The first row (of the tree) should contain exactly one 'Z' and it should be placed exactly at the midpoint of the flag width.
    - The second row should contain 2 'Z's more than the first row.
    - The third row should contain 4 'Z's more than the second row
    - The forth row should contain 2 'Z's more than the third row
    - The fifth row should contain 4 'Z's more than the fourth row
    - etc.
5. The last three rows in the tree are always composed of 3,5, and 7 'Z's regardless of the flag dimensions

Write a java program, `AsciiArt.java`, that takes the number of rows as input from command line and outputs the appropriate ASCII Art picture of the Lebanese flag.

Problem 2 – Bowtie

Write a method, `bowtie`, that takes an integer height as a parameter. It then uses this parameter to draw and appropriately size the figure below. Assume that the height is always an odd number greater than 5. The following pictorial illustrates the figure details when height is equal to 7:



Write a program, **`Bowtie.java`** that calls the method for different values of height.

Sample Run 1 (when height = 7)

```
C:> java Bowtie
++++++++      ++++++++
+++++++++    +++++++++
++++++++++  ++++++++++
++++++++++++++++++++++
++++++++++  ++++++++++
+++++++++    +++++++++
++++++++      ++++++++
```

Sample Run 2 (when height=15)

```
C:> java Bowtie
++++++++++++++++              ++++++++++++++++
++++++++++++++++++          ++++++++++++++++++
++++++++++++++++++++      ++++++++++++++++++++
++++++++++++++++++++++  ++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++  ++++++++++++++++++++++
++++++++++++++++++++      ++++++++++++++++++++
++++++++++++++++++          ++++++++++++++++++
++++++++++++++++              ++++++++++++++++
```

Problem 3 – Pi for Breakfast

There are these mornings where you really wish you had something sweet to start the day. And what better way to start the day with a pie for breakfast, but not just any pie, we will be having pi. To start off with our lovely breakfast we will be using the famous recipe of G. Leibniz (and others). However this recipe gives us only a slice of the pi, namely $\frac{\pi}{4}$. It appears that if we start summing numbers of the form of $\frac{(-1)^n}{2n+1}$ starting from n = 0 till $n = \infty$ we get $\frac{\pi}{4}$

So we have: $1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}$

More technically the Leibniz series is given by the following: $\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$

Computing such a large sum is very tedious to do by hand (especially in the morning). You need to write a program that estimates the pi value. The program will take as input a positive integer k and will compute the sum from n = 0 till n = k of all the Leibniz Series terms. Then your program should output the estimated value of π. For k = 3 the program will compute:

$$1 - \frac{1}{3} + \frac{1}{5}$$

And then deduce the value of π

Sample Run 1:

```
java PiforBreakfast 600
Pi: 3.139925988081
```

Sample Run 2:

```
java PiforBreakfast 700000
Pi: 3.141591225018
```

Problem 4 - Grid

Write a program, `Grid`, which accepts two integers from the command line; name the first one `rows` and the second `cols`. Pass these two integers to a method named `printGrid` that outputs a comma-separated grid of numbers where `rows` represents the number of rows of the grid and `cols` represents the number of columns. The numbers count up from 1 to (`rows x cols`). The output is displayed in column-major order, meaning that the numbers shown increase sequentially down each column and wrap to the top of the next column to the right once the bottom of the current column is reached. Assume that `rows` and `cols` are greater than 0.

Sample Run 1:

```
java Grid 3 6
1, 4, 7, 10, 13, 16
2, 5, 8, 11, 14, 17
3, 6, 9, 12, 15, 18
```

Sample Run 2:

```
java Grid 5 3
1, 6, 11
2, 7, 12
3, 8, 13
4, 9, 14
5, 10, 15
```

Sample Run 3:

```
java Grid 4 1
1
2
3
4
```

Sample Run 4:

```
java Grid 1 3
1, 2, 3
```

Problem 5 - Digits

In this program, `Digits`, you will have to write three methods and test each one of them by calling them from main.

1. Write a method `numDigits` that takes an integer n and returns the number of digits of n.
   Examples: `numDigits(12)` returns 2 and `numDigits(1642)` returns 4

2. Write a method `flipNum` that takes as input an integer number n of any size (that fits in an integer variable) and returns an integer x such as x is the flipped n.
   Examples: `flipNum(142)` returns 241 and `flipNum(4521)` returns 1254

3. Write a method `palindrome` that takes a number `a` and checks whether it is a palindromic number. A palindromic number is a number that remains the same if read in both directions. Example of palindromic numbers include all numbers between 0 and 9 and numbers like: 151 12321 5672765 323 1331.

Problem 6 – Functions

In this problem, `Functions`, you have to write several methods that interact with each other. You are NOT allowed to use the built-in methods of the `Math` class, for example, `Math.pow(…)`, `Math.sqrt(…)`, etc.

a. Write a method called `factorial` that takes an integer and returns the factorial of that integer. The signature of the method is:

```
double factorial(int)
```

b. Write a method called power that takes two integers (`a` and `b`), computes `a` raised to the power `b`, and returns the result. The signature of the method is:

```
double power(int , int)
```

c. Write a method called `exponentialSum` that takes an integer `n` and computes/returns the following sum:

$$s = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!}$$

Note that when `n` is large enough, `s` gets closer and closer to `exp(1)=2.71…`
Use the methods you did in parts `a` and/or `b` to solve this method

d. Write a method called `cosineSum` that takes an even number `n` and computes/returns the following sum:

$$s = 1 - \frac{\pi^2}{2!} + \frac{\pi^4}{4!} - \frac{\pi^6}{6!} + \cdots - \frac{\pi^n}{n!}$$

Note that when `n` is large enough, `s` gets closer and closer to `cos(π)=-1`
Use the methods you did in parts `a` and/or `b` to solve this method

e. Test the methods you did in parts `c` and `d`, by calling them from `main` and passing them an integer value that was inputted from command line.

Submission Instructions and Notes

- Submit your commented source code in a zip file to Moodle by the deadline.
- Your zip submissions should be named `asst2_`*`netid`*, where *`netid`* stands for your AUBnet user name. For example, if your AUBnetid is abc65, you should call your submission asst2_abc65.