COMP 1406: Pure Puzzles

Putting Problem Solving Techniques Into Practice

Output Patterns

Half of a Square

Using only single-character output statements that output one of `'#'`, `' '`, or `'\n'`, write code to produce the following shape:

```
#####
####
###
##
#
```

A Square

Using only single-character output statements that output one of ‘#’, ‘ ’, or ‘\n’, write code to produce the following shape:

```
#####
#####
#####
#####
#####
```

Reduce the problem

A Line

Using only single-character output statements that output one of ‘#’, ‘ ’, or ‘\n’, write code to produce the following shape:

```
#####
```

Reduce the problem… more!

Sideways Triangle

Using only single-character output statements that output one of `'#'`, `' '`, or `'\n'`, write code to produce the following shape:

```
#
##
###
####
###
##
#
```

Sideways Triangle

We know how to:

- Display a row of symbols with a loop
- Display a series of rows using nested loops
- Create a varying number of symbols in each row using an algebraic expression

Start with what you know

Sideways Triangle

What happens if we subtract each row from a larger number like we did with the half square?

Half square:	(numRows + 1) **-** row
Sideways triangle:	(numRows + 1) – row ??

Sideways Triangle

What happens if we subtract each row from a larger number like we did with the half square?

Row Number	8 - row
1	7
2	6
3	5
4	4
5	3
6	2
7	1

We need to go up first, then down

Sideways Triangle

What happens if we subtract each row from the middle row's number?

Row Number	4 - row
1	3
2	2
3	1
4	0
5	-1
6	-2
7	-3

We don't want negative numbers…

Sideways Triangle

What happens if we subtract each row from the middle row's number?

Row Number	abs(4 – row)
1	3
2	2
3	1
4	0
5	1
6	2
7	3

Close, but isn't this the opposite of what we want? It's the number of spaces at the end of the row!

Sideways Triangle

Subtract the number of spaces from the largest row length.

Row Number	4 - abs(4 – row)
1	1
2	2
3	3
4	4
5	3
6	2
7	1

Input Processing

Luhn Checksum

The Luhn formula is a widely used system for validating identification numbers. Using the original number, double the value of every other digit, starting with the rightmost one. Then add the values of the individual digits together (if a doubled value now has two digits, add the digits individually). A check digit is then added to the sum. The identification number is valid if the final sum is divisible by 10.

Poll Everywhere Question

The Luhn formula is a widely used system for validating identification numbers. Using the original number, double the value of every other digit, starting with the rightmost one. Then add the values of the individual digits together (if a doubled value now has two digits, add the digits individually). A check digit is then added to the sum. The identification number is valid if the final sum is divisible by 10.

Given the following identification number, what should the check digit be so the number is valid?

ID number: 657613

Text: 37607

1005680: 6
1005681: 7
1005682: 8
1005683: 10

Luhn Checksum Validation

Write a program that takes an identification number (including its check digit) of arbitrary length and determines whether the number is valid under the Luhn formula.

Luhn Checksum Validation

Issues we need to tackle:

- Knowing which digits to double
- Treating doubled numbers 10 and greater according to their individual digits
- Knowing we've reached the end of the number
- Reading each digit separately

Break the problem down, make a plan

Step 1: Doubled Digits Larger than 10

What is the range of possible values?
What does this mean in terms of processing numbers for the sum?

Step 2: Reading Digits

Can we read the number into an `int`?
How do we get the numeric value of each digit?

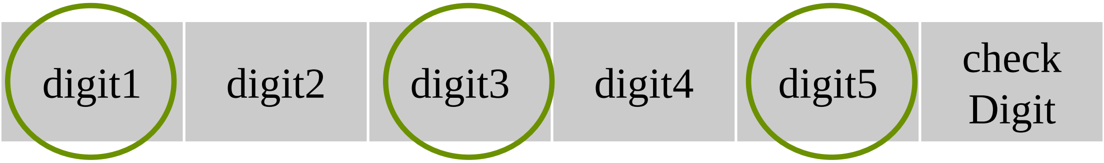
Step 3: Luhn Checksum, Fixed Length

Write a program that takes an identification number (including its check digit) of **length six** and determines whether the number is valid under the Luhn formula. The program must process each character before reading the next one.

Reduce the problem

Step 3: Luhn Checksum, Fixed Length

Write a program that takes an identification number (including its check digit) of **length six** and determines whether the number is valid under the Luhn formula. The program must process each character before reading the next one.



digit1 digit2 digit3 digit4 digit5 check Digit

Step 4: Luhn Checksum, Even Numbers of Arbitrary Length

How would you handle even numbers of arbitrary length?

Step 5: Luhn Checksum, All Numbers of Arbitrary Length

How can we handle both even and odd numbers when we can't know which the number will be until we've processed all the characters?