# Name: Mark: \_\_\_ / 50 = \_\_\_\_ %

# Understanding computer memory

**Question 1: How old is your CS teacher? ( \_\_ /3 marks)**

Add these two binary numbers:

11011 + 1110

|  |
| --- |
|  |

Convert that answer to decimal to see how old your Computer Science teacher is.

Show all your steps to receive full marks

|  |
| --- |
|  |

**Question 2: Hollywood really bytes ( \_\_ /4 marks)**

Hollywood movies are filmed at about 30 frames per second.

Assuming a movie lasts 2 hours, determine the total number of frames in the whole movie. Show your calculations to get full marks.

|  |
| --- |
|  |

Digital movies take up about 3 gigabytes of memory on your hard drive (that’s about 3,000,000,000 bytes).

Use this fact and your answer above to estimate **how many bytes of memory each frame of a movie takes up.** Show your calculations to get full marks.

|  |
| --- |
|  |

**Question 3: The most boring text file of all time ( \_\_ / 10 marks)**

Make a folder on your desktop called ***Boring Files***.

Put the following files into the folder (they’re on Edmodo)

* *One character.txt*
* *Twenty five thousand characters.txt* ( a file that contains 25,000 letter a’s.)
* *One hundred thousand characters.txt* ( a file that contains 100,000 letter a’s.)

Open the folder and notice how big each file is, as measured in bytes.

Use this information to estimate how much memory a single text character takes up. Fill in the chart below to help you make your estimate.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File name** | **Number of a’s in it** | **The size of the file in kilobytes** | **How many bytes is that?** | **Number of bytes divided by the number of characters** |
| *One character.txt* |  |  |  |  |
| *Twenty five thousand characters.txt* |  |  |  |  |
| *One hundred thousand characters.txt* |  |  |  |  |

**Conclusion 1**: In a large text file, a single character takes up about \_\_\_\_\_\_ bytes.

**Question 3a**: How long do you think Mr. S. spent typing out all those “a”s?

|  |
| --- |
|  |

**Question 4: Introducing *hexadecimal*, the numbering system with   
 MORE than 10 digits! ( \_\_ /4 marks)**

Binary digits have two possible values: 0 and 1

Decimal digits have ten possible values: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Hexadecimal digits have 16 possible values: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

where A = 10, B = 11, C = 12, D = 13, E = 14 and F = 15.

Example: The hexadecimal number **24D** converted to decimal would be

**2** x 162 = 2 x 256 = 512

**4** x 161 = 4 x 16 = 64

**D** x 160 = 13 x 1 = 13

512 + 64 + 13 = 589

**Boring fact: *Binary numbers are used to represent text characters in ASCII, while hexadecimal numbers are used to represent text characters in Unicode***. *(Toss that fact around at a party sometime, and then watch how the crowds gravitate towards you as they wallow in the grandeur of your knowledge.)*

Now back to the assignment.

Convert the hexadecimal number **FA7** to decimal. Show all steps.

|  |
| --- |
|  |

Check your answer using the conversion tool at

<http://www.mathsisfun.com/binary-decimal-hexadecimal-converter.html>

Because hexadecimal digits range in value from 0-15, how many binary bits would be required to store a 1-digit hexadecimal number?

|  |
| --- |
|  |

**Question 5: The incredible sweetness of Unicode ( \_\_ /14 marks)**

The ASCII encoding system was a great invention for its time, but it’s starting to reach the end of its usefulness.

Let’s see why.

Each ASCII text character, such as **“h”** or **“@“**, is represented by a byte (that’s 8 bits). (For instance, **“h”** is encoded as **“01101000”** and **“@”** is encoded as **“01000000”**.) This means that the ASCII system can only represent up to

|  |
| --- |
|  |

different characters. That pretty much limits us to standard keyboard characters.

By contrast, the Unicode encoding system represents text characters using

up to 8 hexadecimal digits (though most characters use only 4.)

For instance, the smiley face ☺ is encoded as **“263A”** and a medieval-font   
capital H (**ℌ)**is encoded as “**210C”.**

To store a Unicode character in memory using 4 hexadecimal digits, a computer would need

|  |
| --- |
|  |

binary bits.

Which means that the Unicode system can represent up to

|  |
| --- |
|  |

different characters (assuing we’re only using just 4 hexadecimal digits)

In theory, Unicode characters can use up to 31 binary bits, which can encode up to

|  |
| --- |
|  |

different characters‼! (In practice, only about a million of these are actually used.)

Wow! Type “wow” in the box below if you agree that this is amazing.

|  |
| --- |
|  |

**Did you know that Python knows Unicode?** Try it yourself! Try this in Python:

**print**( **“\u263a”** )

Does it smile at you?

Try **“\u210C”** and see if you get the cool H.

What math symbol is 221A?

|  |
| --- |
|  |

What Chinese character is D5C9? (If you can’t read Chinese, just copy & paste it in)

|  |
| --- |
|  |

If you thought THAT was cool, just wait til you try this! Paste this program into Python and push F5.

**from tkinter import \***

**from time import \***

**from math import \***

**from random import \***

**master = Tk()**

**screen = Canvas( master, width = 800, height = 800, background = "yellow")**

**screen.pack()**

**screen.create\_text( 400, 400, text="\ud5c9", font="Times 100")**

**screen.update()**

Describe what you see:

|  |
| --- |
|  |

**Question 6: A little programming challenge ( \_\_ /15 marks)**

In this exercise, you will write a Python function called **decimalToBinary** that takes an integer as an argument and returns that number written in binary.

Example:

**print**( decimalToBinary( 17 ) ) would return **“10001”**

To get started:

On paper, convert 39 to binary using the method we learned in class yesterday.

As you’re doing it, ask yourself:

- When does the process stop?

- What kind of loop is this?

Hint: Remember that we use the + sign to join two strings in Python:

X = **“what’s**”

Y = **“up?”**

Z = X + Y would evaluate to the string **“what’s up?”**

**Once you’ve finished your program, paste it here:**

|  |
| --- |
|  |

**BONUS!: (up to 10 bonus marks)**

Write a Python program callled **binaryToDecimal** that converts binary to decimal.

Example:

**print**( binaryToDecimal(“**1110”** ) ) would return **14**

Paste your finished Python program here:

|  |
| --- |
|  |