Worksheet: 10a

"Binary Invaders"

By far the most important and indeed the most common part in a computer is called a **transistor**. A transistor is a tiny electronic switch that can be set to **on** or **off** (**1** or **0**, or **true** and **false**) using electrical charges. The computer's **processor** is made up of **billions** of these tiny **on**/**off** switches.

If a computer only understands **1**'s and **0**'s how does it send emails, play games, make music, show movies and surf the web?

The Binary number system (or **base 2** as it is also called) is based on **1**'s and **0**'s which makes it perfect for computers. Rather than counting in tens, hundreds and thousands it is based on the **power of 2**. Each binary digit from **right to left** increases by the power of 2, creating the pattern **1**, **2**, **4**, **8**, **16**, **32**, **64, 128**.

Each binary digit is called a **BIT**. Binary numbers are usually read in blocks of eight **BITs** at a time called a **BYTE**. **BITs** and **BYTES** are the **lifeblood** of all computers. From music to pictures, from maths to documents, everything in a computer is stored and processed in **BITS** and **BYTES**.

Take the letter '**A**' for example, what exactly is the letter '**A**' to a computer? Computers use an index of numbers to define all the letters and characters used in a font. The letter 'A' for example is **65**. Very confusing yes? Well it gets worse!

A computer only understands binary so how do we express **65** as a binary number? **01000001** that's how!

128	64	32	16	8	4	2	1
0	1	0	0	0	0	0	1

= 65 = A

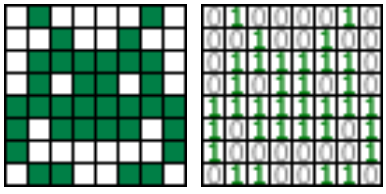
If all the **BITs** are **on** (**1**'s), and you add them all together you have a decimal value of **255**. 1+2+4+8+16+32+64+128 = **255**. There's still one more value though and that's if all the **BITs** are **off** (**0**'s). This of course adds up to the decimal value **0**. In total, we have **256** possible values for an **8 BIT** number.

Enough already - lets have some fun!

```
CLS
DEFCHAR (1, 0, 66, 36, 126, 90, 255, 189, 129, 102, 0)
PRINTAT (0,0); CHR$(1);
END
```

Take a Space Invader...

This is how it might look on screen (**left**) but to the computer it is just a series of switches left in an **on** or **off** position (**right**)



```
CLS
DEFCHAR  (1, 0, 66, 36, 126, 90, 255, 189, 129, 102, 0)
DEFCHAR  (2, 0, 24, 36, 126, 90, 255, 255, 66, 60, 0)
FONTSIZE ( 10 )
INK = Green
LOOP
PRINTAT (0,0); CHR$(1)
WAIT (0.2)
PRINTAT  (0,0); CHR$(2)
WAIT  (0.2)
REPEAT
END
```

The humble 'SPACE INVADER' can be expressed in binary, as can just about any shape, sound or text. **RUN [F3]** the program on the left.

Using the **DEFCHAR** command we can redefine a character to look like a Space Invader. In binary it looks like this;

128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	0	= 0
0	1	0	0	0	0	1	0	= 66
0	0	1	0	0	1	0	0	= 36
0	1	1	1	1	1	1	0	= 126
0	1	0	1	1	0	1	0	= 90
1	1	1	1	1	1	1	1	= 255
1	0	1	1	1	1	0	1	= 189
1	0	0	0	0	0	0	1	= 129
0	1	1	0	0	1	1	0	= 102
0	0	0	0	0	0	0	0	= 0

Enter the second program on the left and **RUN [F3]** to see what happens. Cool eh?

Worksheet: 10b

"Binary Invaders"

```
CLS
DEFCHAR (1, 126, 126, 24, 24, 24, 24, 24, 24, 126, 126)
DEFCHAR  (2, 0, 36, 126, 255, 255, 255, 126, 60, 24, 0)
DEFCHAR  (3, 0, 66, 36, 126, 90, 255, 189, 129, 102, 0)
FONTSIZE ( 8 )
LOOP
INK = Yellow
PRINTAT (0,0); CHR$(1)
WAIT (0.5)
INK  = Pink
PRINTAT  (0,0); CHR$(2);
WAIT  (0.5)
INK  = Green
PRINTAT  (0,0); CHR$(3);
WAIT  (0.5)
REPEAT
END
```

HACKER CHALLENGE:
Change the colours and the delays to make the animation show faster or slower.

You could make the character even larger too (FONTSIZE).

Change one, or all of the characters for ones you have designed yourself on graph paper. You will need to replace the numbers in the DEFCHAR command but remember, the first number tells the computer which character to replace.

If you want to add more then you will need to increase the first number in the sequence.

DEFCHAR (1, ##, ##, ##, ##, ##, ##, ##, ##, ##, ##)

Finally, can you change the position on the screen of the character?

Experiment to see how changing the values in PRINTAT (0,0) make a difference to its location.

NOTE: The first number changes the horizontal position (x) and the second number is the vertical (y). The characters don't have to be in the same position each time. They could even be random!

Edit your code so it is as displayed on the left then **RUN [F3]**

Notice the first digit in the **DEFCHAR** sequence. This corresponds to the number in the **CHR$(#)** command. We are redefining **CHR$(1)**, **(2)** & **(3)** in the computers memory.

We now have three characters, and are animating them in sequence, to tell a story.

Please grab some graph paper, a pen and pencil and, if you are not confident with addition, maybe a calculator too.

Draw the following grid pattern, with a pen, with the numbers along the top. You need 8 columns across and 10 rows down.

If you have a **FUZE** Binary notepad, use that!

128	64	32	16	8	4	2	1	
								=
								=
								=
								=
								=
								=
								=
								=
								=
								=

Then, using a pencil, lightly fill in boxes to draw your own characters, shapes or symbols.

Add up the values of each box in a row to get to your total figure at the end.

Now copy these into the **DEFCHAR** statements to define the character. Just replace the ones already there or create new ones where:

The first number in the **DEFCHAR (#)** sequence specifies the character number that you use in the **PRINT CHR$ (#)** command.

Be creative, although it is not easy with a such a small number of pixels to work with!