1.

For each of the following C++ expressions, write its value and its type.

For example, for part (a) you should write **3** and **int**.

(a)   `1 + 2`

(b)   `26 % 10`

(c)   `56 / 5`

(d)   `8 * 7 / 5`

(e)   `(8 * 7) / 5`

(f)   `8 * (7 / 5)`

(g)   `8 / 5 * 7`

(h)   `8.0 / 5 * 7`

(i)   `8 / 5.0 * 7`

(j)   `8 / 5 * 7.0`

(k)   `1 / 2 + 8 / 4`

(l)   `(1 / 2) + (8 / 4)`

(m)   `(1 / 2) + (8.0 / 4.0)`

(n)   `1234 - 1234 * 10 / 10`

(o)   `1 / 2 > 0.333`

(p)   `1 * 2 * 3 * 4 / 1 * 2 * 3 * 4`

2. [For this question, negative numbers are of no interest]

(a)

Examine this function:

```
void tom(int a)
{ if (a==0)
     cout << "x";
   else
   { tom(a-1);
     tom(a-1); } }
```

What value for the parameter `a` would make it easiest to analyse?
    and what would `tom` do, given that simplest value?
What parameter value becomes very simple to analyse once you know that?
    and what would `tom` do, given that value?
What parameter value becomes very simple to analyse once you know that?
    and what would `tom` do, given that value?

Now work it out. State in plain English what the `tom` function does.

(b)

Now analyse this function:

```
int jerry(int z)
{ if (z==0)
     return 1;
   else
   { const int p = jerry(z-1);
     return p*3; } }
```

What, in plain English, does `jerry` calculate?

(c)

This is a quick little question. Just look at the function, it doesn't need complex analysis. The function is supposed to print a count-down of numbers starting from N. What is wrong with it?

```
int spike(int N)
{ if (N>0)
   cout << N << "\n";
   spike(N-1); }
```

3.

(a)

Without using loops or variables:
write a C++ function that takes two int parameters, a and b, and prints the cubes of all the numbers between a and b inclusive.

example: `parta(2, 9)` should print `8 27 64 125 216 343 512 729`

(b)

Without using loops or variables:
write a C++ function that takes two int parameters, a and b, and prints the cubes of all the numbers between a and b inclusive, only if the last digit of the cube is between 2 and 5.

example: `partb(2, 9)` should print `64 125 343 512`

(c)

Without using loops or variables:
write a C++ function that takes two int parameters, a and b, and returns as its result how many numbers between a and b have a cube whose last digit is between 2 and 5.

example: `const int N = partc(2, 9);` should set N to 4 because `partb(2, 9)` printed four numbers.

Note
If you really can't answer this question without using loops or variables, then answer it with loops and variables, but it will only be for very partial credit.

4.

Draw a picture of your favourite finger.