C PROGRAMMING: DO-WHILE LOOP

A for loop is a useful way to get a computer to do a task a known number of times. As an example, for(j=1; j⟨N; j=j+1) {$\cdots$} sets j initially to 1, changes j by 1 each time the lines {$\cdots$} within the loop are executed, and then stops when $j = N$.

do-while loops are another way to get the computer to do tasks over and over, and are especially useful when you want to stop based on some condition occurring and you don't really know how many repetitions it will take.

Probably best to skip the first example and go straight to the Collatz problem of page 2.

We first illustrate a do-while loop where the stopping condition is obvious (so this task could just as easily be done with a for loop.

```
#include <stdio.h>
#include <math.h>
int main(void)
{
int j=43;
do
{
j=j-1;
printf("\n%i",j);
}
while(j>0);
return 0;
}
```

Comments:

[1] This is very simple example where it is clear this do-while loop will execute 43 times.

The code below is a better illustration of how a do-while loop differs from a for loop.

```
#include <stdio.h>
#include <math.h>
int main(void)
{
int j;
printf("Please enter initial integer\n");
scanf("%i",&j);
do{
if (j%2==0) {j=j/2;}
else
{j=3*j+1;}
printf("%i\n",j);
}while(j!=1);
return 0;
}
```

[2] The above code tests the Collatz conjecture which states if you start with any integer and divide by two if it is even, and multiply by three and add one if it is odd, and repeat over and over, you will always end up at one.

[3] A do-while loop is perfect for a code testing the Collatz conjecture, because you have no idea how many iterations it will take. You only know the stopping condition (reaching one).

[4] The Collatz conjecture, an **unproven(!!)** statement in mathematics. Want to be famous? Then prove it!