Skel Tutorial

Given the following C code, can you generate the corresponding Skel?

```
#include <stdio.h>
#include “some_other_header.h”

int returnFour()
{
	int tmp;

	return tmp;
}

double gimmeFive(int a, double b)
 {
	float tmp;

	return tmp;
}

int main(int argc, char** argv)
{
	printf(“\nHello, Master!\n\n”);

	return 0;
}
```

First, let us begin with the libraries up at the top, “stdio.h” and “some_other_header.h”; from the use of the inequality symbols for “stdio.h” and the quotation marks for “some_other_header.h”, we ascertain that the first is a standard library, and the second is a custom one. Recall that standard library includes in Skel take the form:

```
#: std_lib_name
```

Custom library includes take the form:

```
!: cst_lib_name
```

So, we are left with

```
#: stdio.h
!: some_other_header.h
```

Remember, these do not have to be at the top of Skel code.

Next, functions; Recall that the abbreviated (and most convenient) form for a function

declaration in Skel is:

@ fn_name, fn_type, (fn_params):;

Further, recall the types:

%i for int
%d for double
%f for float
%c for char
nil for void

Which yields, for function declarations,

@ returnFour, %i, ():;

and

@ gimmeFive, %d, (%i a, %d b):;

Being that there is no need to define main in Skel, and that the above C corresponds to the standard main provided, the full Skel specification for the C code above comes out to:

#: stdio.h ←Can be on any line
!: some_other_header.h ← Can be on any line
@ returnFour, %i, ( ):;
@ gimmeFive, %d, (%i a, %d b):;

or

@ returnFour, %i, ( ):;
#: stdio.h ←Can be on any line
@ gimmeFive, %d, (%i a, %d b):;
!: some_other_header.h ← Can be on any line

or even

@ returnFour, %i, ( ):;
@ gimmeFive, %d, (%i a, %d b):;
#: stdio.h ←Can be on any line
!: some_other_header.h ← Can be on any line

As you can see, the C skeleton above, numbering 23 lines (formatted nicely) is defined by Skel in 4; more than 5 times less lines. Huzzah!

As an exercise, generate the C code from the following Skel:

```
@ practice_fn, nil, ( ):;
#: stdio.h
!: practice.h
!: practice2.h
@ someStuff, %c, (%d a, %c v, %i p):;
#: string.h
```