C Hints

These hints relate to the practical and show the simplest (not necessarily most efficient/shortest code) way of doing various things in C that are needed for the practical. If you are proficient in C, feel free to use other methods.

Copying elements from array B to array A (same size arrays):

```
#define ELEMENTS 4 /* these arrays have 4 elements */
```

```
for(i=0; i&ltELEMENTS; i++) A[i] = B[i];
```

- Note 1: do not use sizeof to determine the number of elements in an array, unless you know what you are doing. It will only give you the number of elements for certain data types. sizeof returns the amount of memory, in bytes, not the number of elements.
- Note 2: you can not assign arrays. ie you can not say A=B;

Structures - `struct`

Structures are a datatype that can hold different types of data inside. If you like, you can think of them as being a bit like an object in Java that only holds public state (no methods).

In this practical, you will be working with either:

- structures - `struct rtpkt packet;`
  or
- pointers to structures - `struct rtpkt *packet;`

To access or assign elements in a structure, use the "dot" notation:

```
packet.sourceid = 2; if (packet.sourceid != 0)....
```

To access or assign elements through a pointer to a structure, use the "arrow" notation:

```
packet->sourceid = 2; if (packet->sourceid != 0).....
```

Note 1: when calling functions, you need to pass the expected type. For example `tolayer2` expects a struct rtpkt

If you have a struct and need a pointer to a struct, put '&' in front of it

```
some_function(&some_struct);
```

If you have a pointer and the function wants the structure itself, put a '*' in front of it

You shouldn't need to do this in this prac, but in case you want to:

```
some_function(*ptr);
```

That's it. Everything else is loops, if conditionals and basic types and assignments, which have the same syntax as in Java and C++.

Compiling a C program

You can compile a program called myprogram.c by typing:

```
gcc myprogram
```

this will produce an executable called a.out If you want to call the executable program something else, use:

```
gcc myprogram.c -o <executable name>
```

where <executable name> is the name you want to use.