Problem 1

A phone company stores information on one phone call in *one* 64-bit variable of type **unsigned long long** (or, even better although usually equivalently, **uint64_t**) which contains:

1. identifier of the calling customer (caller): 17-bit number, i.e., from interval $[0, 2^{17} - 1] = [0, 131071]$;
2. zone number of the caller (caller_zone): 7-bit number, i.e., from interval $[0, 2^7 - 1] = [0, 127]$;
3. identifier of the receiving customer (callee): 17-bit number;
4. zone number of the callee (callee_zone): 7-bit number;
5. duration of the call in seconds: 13-bit number, i.e., from interval $[0, 2^{13} - 1] = [0, 8191]$;
6. tariff number: 3-bit number, i.e., from interval $[0, 2^3 - 1] = [0, 7]$;

Write functions

- **encode** — taking six numbers described above (as **int**s) and packing their values into *one* number of type **uint64_t**;
- **info** — printing information on one phone call passed to it as a single number of type **uint64_t**.

For example, the program

download *Telephones.cpp*

```cpp
#include <iostream>

uint64_t encode(int caller,   int caller_zone,
                int callee,   int callee_zone,
                int duration, int tariff) {
    // ...
}

void info(uint64_t res) {
    // ...
}

int main () {
    info(encode(130999,101,7777,99,7000,6));
}
```

should print

```
Caller:      130999
Caller_zone: 101
Callee:      7777
Callee_zone: 99
Duration   : 7000
Tariff     : 6
```

Do *not* use any tools from the standard library (except those in ***iostream*** for outputting the results). Do *not* use bit fields, just bit operations like ANDing, ORing, shifting...
