Manual reduction

1 Manual reduction

This exercise is about parallelizing a reduction operation on all the elements of an integer array `x` of size `n`. Consider a binary operator $\otimes$ which is **associative** and **commutative** which means

$$a \otimes b \otimes c = (a \otimes b) \otimes c = a \otimes (b \otimes c) = (b \otimes c) \otimes a = ...$$

In the provided sequential code, this operator is implemented by the `int operator(int a, int b)` function which returns `a`$\otimes$`b`. Furthermore, the overall result of the reduction must be initialized to identity value of operator $\otimes$; this can be done through the `int init()` routine. The provided sequential reduction code simply performs a sweep of the whole array `x`:

```
res = init();
for(i=1; i<n; i++)
  res = operator(res, x+i);
```

and thus stores in `res` the result of (((`x[0]`$\otimes$`x[1]`)$\otimes$...)$\otimes$`x[n-1]`).

2 Package content

In the `manual_reduction` directory you will find the following files:

- `main.c`: this file contains the main program that creates a vector x of size n and then computes its reduction using first the sequential routine `sequential_reduction` and then the parallel routine `parallel_reduction` which has to be implemented as described below. **Only the `parallel_reduction` routine has to be modified for this exercise**.
- `aux.c, aux.h`: these two files contain auxiliary routines and **must not be modified**.

The code can be compiled with the `make` command: just type `make` inside the `manual_reduction` directory; this will generate a `main` program that can be run like this:

```
$ ./main n
```

where n is the size of the vector whose reduction has to be computed.

3 Assignment

- At the beginning, the `parallel_reduction` is a copy of `sequential_reduction`. Modify this routine in order to parallelize it. Make sure that the result computed by the two routines (sequential and parallel) is consistently (that is, at every execution of the parallel code) the same.