Introduction to High-Performance Computing

Supplemental document

We assume that a `long` is a 64 bit integer.

```
/*
 * Predefined functions.
 * Their execution time does not depend on the value of their arguments.
 * T is any type, which stands for the type of a sequence element
 */
long aux_size();                        /* Returns N, the total size of the aux array */
T* load_aux(long offset, long size);    /* Returns aux[offset : offset + size] */
T initialize(long j);                   /* Returns the 1st term of the jth sequence  */
T update(T x, T aux);                   /* Computes the next term of the sequence */
bool is_special(T x);                   /* Determine whether a term is special */

/*
 * Iterates the jth sequence until the Nth term, and determines whether the latter is special
 */
bool check(long j, long N, const T * aux)
{
    T x = initialize(j);          /* x_0 */
    for (long i = 0; i < N; i++)
        x = update(x, aux[i]);    /* x_{i+1} = F(x_i, a_i) */
    return is_special(x);
}

/*
 * Tries all sequences until finding one which is special.
 * Returns an integer j such that check(j, N, aux))
 */
long find(long N, const T * aux)
{
    long j = 0;
    while (true) {
        if (check(j, N, aux))
            return j;
        j += 1;
    }
}

/*
 * Sequential program that must be parallelized
 */
int main()
{
    long N = aux_size();
    T * aux = load_aux(0, N);
    printf("résultat : %ld\n", find(N, aux));
}
```