There are two packages in this project
(1) core: synchronization framework(implementation and interface) + compute framework (interface only)
(2) myuts: UTS implementation based on GLB (aka core)
[**note:**
(1)the simple build.sh script to compile the project and a binary MyUTS will be generated, one can use it exactly the same way as using Olivier's UTS code)
(2) Red color means what the user of this framework should implement (aka contract)
(3) Blue color shows the concrete UTS example of using/implementing the framework
]
Specifically
core:
T: Type of input data, Z: Type of output data
(1) TaskFrame[T, Z]:

```
    runAtMostNTask(n:Long) // run at most N tasks this is the work
    getResult():Z;// sequentially, one can do while(runAtMostNTask()); getResult
    initTask():void; // init the task, only the first task frame should do this (aka root task)
    getTaskBag():TaskBag[T]; // return the taskbag
```

(2) TaskBag[T]: datastructure that holds the task

```
    merge(tb: TaskBag[T]):void; // merge the incoming task bag
    split():TaskBag[T]; // split a task bag, the contract is if it returns null it means it
                        // is not  worth splitting
    size():Long; // return the size of the task bag
```

(3) GlobalJobRunner[T,Z]:

```
    getResultReducer():Reducible[Z]; // get the reducer
    setResultReducer(r:Reducible[Z]):void; // set the reducer
```

(4) LocalJobRunner[T,Z]:

    where synchronization and distribution work is done. Transparent to users.

myuts:
Implement UTSTaskFrame, UTSTaskBag, MyUTS(aka UTSGlobalJobRunner), UTSResultReducible (aka Reducer that can reduce the results)

[**note:**
(1)UTSTree, Queue, Fragment are all the auxiliary classes that facilitate UTSTaskFrame and UTSTaskBag, these three auxiliary classes can be merged to one class when beautifying code.
(2) UTSTreeNode is just a place holder, it doesn't do anything now.]

After all the above things in red are implemented, to run the code, write a main method in your own GlobalJobRunner (in this case MyUTS) following this pattern (see main method in MyUTS, 3 lines of code):

```
val myuts: MyUTS = new MyUTS();   // create a new GlobalJobRunner
myuts.setResultReducer(new UTSResultReducible());  // set the result reducer
result:Long = myuts.main(()=>new LocalJobRunner[UTSTreeNode, Long](new
UTSTaskFrame(b,r,d), n, w, l, z, m));  // call the main method of GlobalJobRunner
```