Haskell Functions

`map`	`map (+1) [1..5]`
`filter`	`filter even [1..5]`
`take`	`take` 2 [-2..10]
`drop`	`drop 1 [0,1..5]`
`head`	`head [1..5] = 1`
`tail`	`tail [1..5] = [4..5]`
`all`	`all even [2,4,6]`
`any`	
`iterate`	
`repeat`	
`zip`	`zip [1..5] [5..9]`
`elem`	`elem 4 [1..5]`
`[(a,b) | a <- [1..5], b<- [5..9]]`	
`(+1) 5`	`=6`
`(\x ->4) 6`	`=6`
`(\x y ->x+y)`	`=5`

2 3

Church Encoding

CTrue	`\t. \f.  t`
CFalse	`\t. \f.  f`
C2	`\s. \x. s (s x)`
CNot	`\a.  a CFalse CTrue`
CAnd	`\a. \b. a a b`
COr	`\a. \b. a b a`
CEven	`\n. n not CTrue`

`

MPI

MPI_comm_sizze(-wrld, &out)	get size
MPI_comm_ran-k(wrld, &out)	get rank
MPI_Scatter(data, amount, type, &local, , amount,..)	
MPI_reduce	a0=a0+a1+a2
MPI_allreduce	a0=sum(); a1=sum-()...
MPI_scatterV(data, [amounts],...	Variable amounts
MPI_gatherV(data, [amounts],...	Variable amounts
MPI_ReduceScatter	Reduce to root then scatter to all
MPI_Alltoall	Transpose Data matrix
MPI_allGather	a=b=c