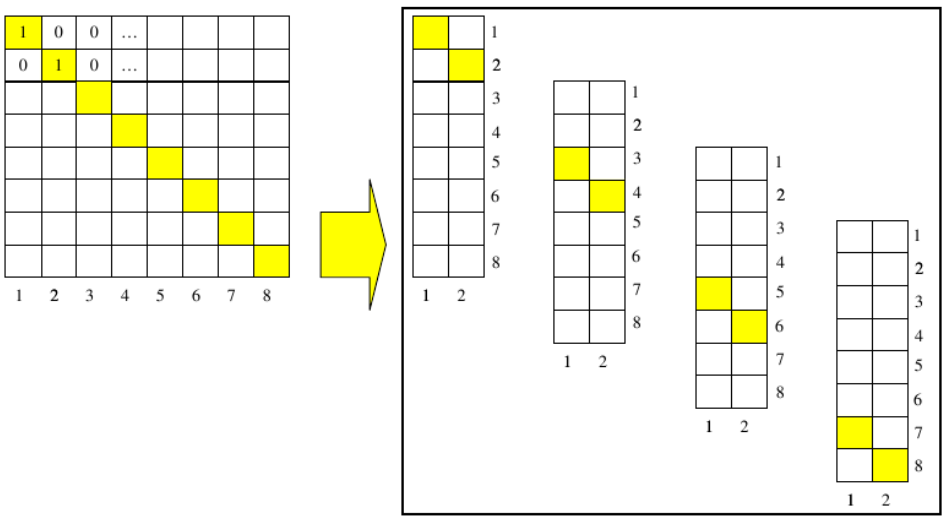
Exercise 6

Exercise 6

Write a program that performs a data distribution over the processes of the identity matrix, as pictured in the following figure:



Given the number of processes and the dimension of the identity matrix, each process must allocate its own portion of the matrix. Depending on the language of your choice, distribute the matrix by rows (C) or by columns (FORTRAN).

First, obtain the number, N, of rows (columns) you need to allocate for each process, np. In case N is not divisible by np, take care of the remainder with the module operator (function).

Now you need to implement a transformation from **global to local (task) coordinates**. You can use a a varable, iglob, that shifts the values to be initialized to 1 according to the global coordinates:

task	iglob (N = 8)	iglob (N = 10)
0	1, 2	1, 2, 3
1	3, 4	4, 5, 6
2	5, 6	7, 8
3	7, 8	9, 10

To check if everything is correct make the process 0 collect all matrix blocks initialized by the other processes and print out the matrix:

```
1 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1
```

HINTS:

	C
MPI_SEND	`int MPI_Send(void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)`
MPI_RECV	`int MPI_Recv(void* buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)`
MPI_INIT	`int MPI_Init(int *argc, char ***argv)`
MPI_COMM_SIZE	`int MPI_Comm_size(MPI_Comm comm, int *size)`
MPI_COMM_RANK	`int MPI_Comm_rank(MPI_Comm comm, int *rank)`

MPI_FINALIZE `int MPI_Finalize(void)`

FORTRAN

MPI_SEND
```
MPI_SEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, IERROR)
<type> BUF(*)
INTEGER COUNT, DATATYPE, DEST, TAG, COMM, IERROR
```

MPI_RECV
```
MPI_RECV(BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, STATUS, IERROR)
<type> BUF(*)
INTEGER COUNT, DATATYPE, SOURCE, TAG, COMM, STATUS(MPI_STATUS_SIZE), IERROR
```

MPI_INIT
```
MPI_INIT(IERROR)
INTEGER IERROR
```

MPI_COMM_SIZE
```
MPI_COMM_SIZE(COMM, SIZE, IERROR)
INTEGER COMM, SIZE, IERRO
```

MPI_COMM_RANK
```
MPI_COMM_SIZE(COMM, SIZE, IERROR)
INTEGER COMM, SIZE, IERROR
```

MPI_FINALIZE
```
MPI_FINALIZE(IERROR)
INTEGER IERROR
```

‹ Solution 5 up Solution 6 ›