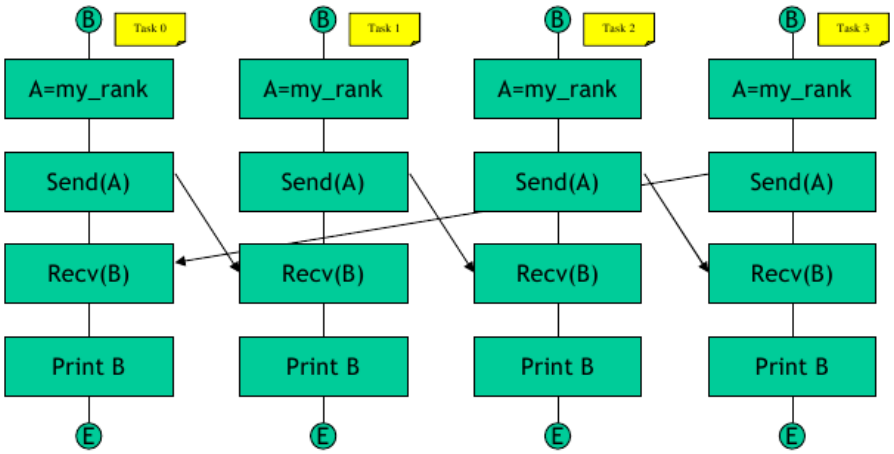
Exercise 3

Exercise 3

Write a code that using point to point communications performs a circular send/receive as represented in the following figure



Each task will declare two float arrays, A and B, of a fixed dimension (10000). Every element of A will be initialized with the rank of the process. Then, A and B will be used as the buffers for SEND and RECEIVE, respectively. Each process sends only to the process on its right and receives only from the process on its left. As you can see from the picture the last process will send its array A to the first task.

Avoid deadlocks and make sure that the code will work with a general number of tasks. Therefore, you need two variables to be used as destination and source for the send and receive calls, inizialized as:

```
        | my_rank + 1      every task except the last
right
  =     |
        | 0
```

last task

left = | my_rank - 1 every task except the first

| last rank first task

The program terminates with each process printing out one element of the array B.

```
I am task 0 and I have received b(0) = 3.00

I am task 1 and I have received b(0) = 0.00

I am task 2 and I have received b(0) = 1.00

I am task 3 and I have received b(0) = 2.00
```

HINTS:

C

MPI_ISEND

```
int MPI_Isend(void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm, MPI_Request *request)
```

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

CODE

right / left	Use the module function "mod" in FORTRAN, or the "%" operator in C

‹ Q/A Exercise 2 up Solution 3 ›