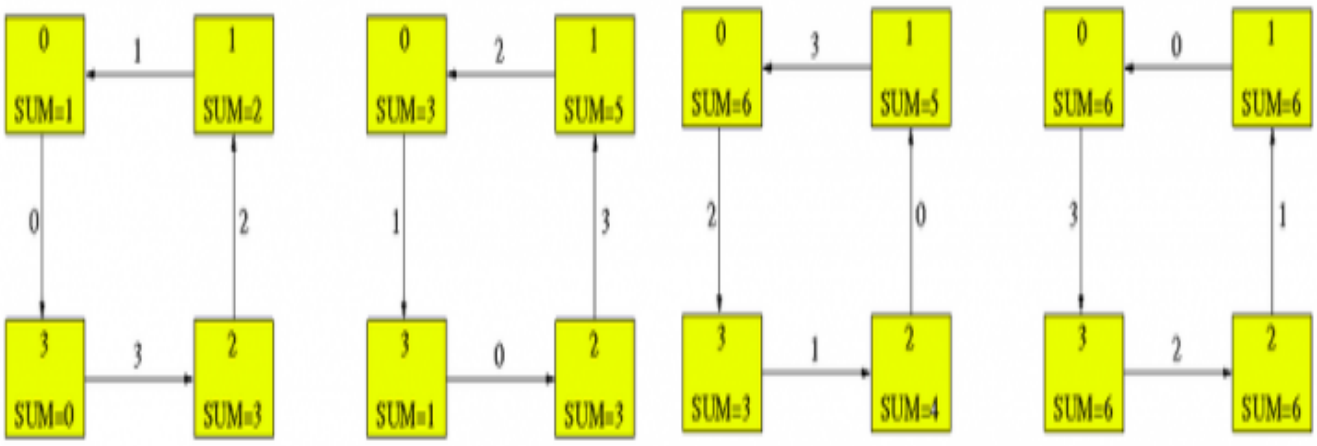
Exercise 4

Exercise 4

Write a code that performs a circular sum of the tasks' ranks. Let A be a float initialized to the rank of the task, and B a float used as a buffer for the receive call. Then, let SUM be a variable for storing the partial sum. The code should execute a series of send/receive calls, and, at each step:

- SUM is updated with the just received buffer B;
- the send buffer A is set to the just received buffer B.

The following figure represents what the code should do



At the end of the execution, all the processes will have stored in their variable SUM the sum of all the ranks.

```
I am proc 0 and sum = 6.00
```

```
I am proc 1 and sum = 6.00

I am proc 2 and sum = 6.00

I am proc 3 and sum = 6.00
```

HINTS:

C

MPI_SENDRECV

```
int MPI_Sendrecv(void *sendbuf, int sendcount, MPI_Datatype sendtype, int dest, int sendtag, void *recvbuf, int recvcount, MPI_Datatype recvtype, int source, int recvtag, 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_SENDRECV

```
MPI_SENDRECV(SENDBUF, SENDCOUNT, SENDTYPE, DEST, SENDTAG, RECVBUF, RECVCOUNT, RECVTYPE, SOURCE, RECVTAG, COMM, STATUS, IERROR)
<type> SENDBUF(*), RECVBUF(*)
INTEGER SENDCOUNT, SENDTYPE, DEST, SENDTAG, RECVCOUNT, RECVTYPE, SOURCE, RECV 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 3 up Solution 4 ›