Assignment 11. MPI. Combined reception and transmission of messages.

Based on Assignment 10, write a program for ring topology exchange using the **MPI_Sendrecv**() function.

In situations where you need to exchange data between processes, it is safer to use the overlaid **MPI_Sendrecv** operation. The **MPI_Sendrecv** function combines the execution of the send and receive operations. Both operations use the same communicator, but message IDs may differ. The location of the received and transmitted data in the address space of the process should not overlap. The data sent can be of different types and lengths.

In cases when it is necessary to exchange data of the same type with replacement of the sent data with the received ones, it is more convenient to use the **MPI_Sendrecv_replace** function. In this operation, the data sent from the buf array is replaced with the received data.

The special address **MPI_PROC_NULL** can be used for **source** and **dest** in data transfer operations. Communication operations with such an address do nothing. The use of this address is convenient instead of using logical constructs to analyze the conditions to send / read a message or not.

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**)

Input parameters:

sendbuf - the address of the data to be sent

sendcount - the number of sent variables

sendtype - the type of data being sent

dest - destination rank

sendtag - the tag of the sent message

recvcount is the number of received data.

recvtype - the type of data being received

source - from whom the message is received

recvtag - received message tag

comm - communicator

Output parameters:

recvbuf - the address of the received data

status - message received status

sendbuf and **recvbuf** must not overlap.